HAPPY-COMPUTER SONDERHEFT

SONDERHEFT 9 ÖS 100,-/SFr. 14,- Lit. 12000/hfl. 18,-/dkr. 68,- DM 14,-

Markt&Technik

68000er




MEGA ATARI
ATARI SF354
ATARI 520ST+
Computer des Jahres in der
PC-Klasse ist der ATARI 520
Der neue 520 ST+, der
MEGA ATARI , ist
stärker. 1 MB steh
im Arbeitsspeiche
zur Verfügung. So
können auch kom
plexe Operationen
zeitsparend durchge
führt und
anspruchsvollste Anw
dungsprogramme, selbst mit hohem Speicherbed
bearbeitet werden. Anwendern bietet ATARI in
PC-Klasse diese Spitzentechnologie zum ungewö
lich niedrigen Preis. Dazu erhalten Sie Software-
programme für alle ATARI PC der ST-Serie beim
Fachhandel mit dem blauen ATARI-Schild an der
1 MB RAM
16/32-bit
ATARI 520 ST+ incl. Monitor SM 124 und Floppy SF 354
DM 2.698,—
unverbindl. Preisempfehlung
ATARI®
... wir machen Spitzentechnologie preiswert.



Wir machen Zukunft

Wir, Leser und Macher der »68000er«, formen gemeinsam ein Stück Zukunft durch unser Interesse und Engagement für eine neue Technologie, die den Bedürfnissen der Zeit entspricht. Eine Technologie der schnellen, speicherstarken, grafiktalentierten, anwenderfreundlichen und bei all den enormen Fähigkeiten auch noch preiswerten Computer mit der 68000-CPU.

Und damit wir, Leser wie Macher dieses Magazins, wissen was in Zukunft Sache ist, haben wir uns im Land der Trends, in USA, umgesehen. Die Ergebnisse sind beeindruckend.

Auf ganz andere Weise beeindruckten uns Video-Digitalisierer. Wie Visionen aus einer anderen Welt wirkten auf uns die Resultate, besonders solche, bei denen Realität und Fantasie ineinanderfließen. Bei unseren Tests entstanden unglaubliche und faszinierende Bilder auf dem Computer.

Mit der Faszination, die von digitalisierten Bildern ausgeht, beschäftigen wir uns ausführlich in dieser Ausgabe.

Aber nicht nur Bilder, sondern auch Töne sind in binäre Zahlenkolonnen umsetzbar und somit vom Computer zu verarbeiten. Wir präsentieren Ihnen deshalb eine detaillierte Bauanleitung eines Sound-Digitalisierers mit komfortabler Software und ausführlichen Grundlagen. Für weniger als 60 Mark und mit etwas Geschick basteln Sie sich Ihren Sound-Digitalisierer für den Atari ST selbst. Die Anwendungen reichen von der akustischen Untermalung von Bildershows über realistische Geräusche bei Spielen bis hin zur Sprachausgabe.

Programmierer brauchen gute Software, um Fantasie in Realität umzusetzen. Wir unterzogen deshalb wieder eine Menge beeindruckender Programme ausführlichen Tests, unter anderem »1st Word plus«. Die Basis dieses Programms, »1st Word«, mauserte sich in kurzer Zeit zum Standard bei Textverarbeitungen für den ST. Eine Anzahl neuer Funktionen, wie Grafikeinbindung und Spellchecker, sichern »1st Word plus« den ersten Platz. Um auch den Entwicklern unter Ihnen gerecht zu werden, paradiert wieder eine stattliche Anzahl Listings:

Als Weiterführung unseres Spielebaukastens in C folgen in dieser Ausgabe 22 brandheiße Routinen für Animation und 16farbige Sprites sowie ein komfortabler Sprite-Editor. Mit diesem feinen Softwarepäckchen wird Spieleprogrammierung auf dem ST zum Kinderspiel.

Zum ersten Male erscheint auch ein Spiel zum Abtippen für den Amiga. Ein Strategie-Spiel mit toller 3D-Grafik und Maussteuerung läßt das Herz eines jeden Amiga-Besitzers höher schlagen.

Unsere Leserservice-Diskette für den Amiga enthält zusätzlich zu den Programmen aus dieser Ausgabe Bild-»Rohmaterial« und Beispielbilder aus den Beiträgen zum Schwerpunkt »Video-Digitalisierung«. Sowie gratis dazu ein passendes Diaschau-Programm (Public Domain) zum Vorführen der Bilder. Wer ein Malprogramm, wie zum Beispiel Deluxe Paint besitzt, kann gleich praktisch ausprobieren, was in den Beiträgen beschrieben wird.

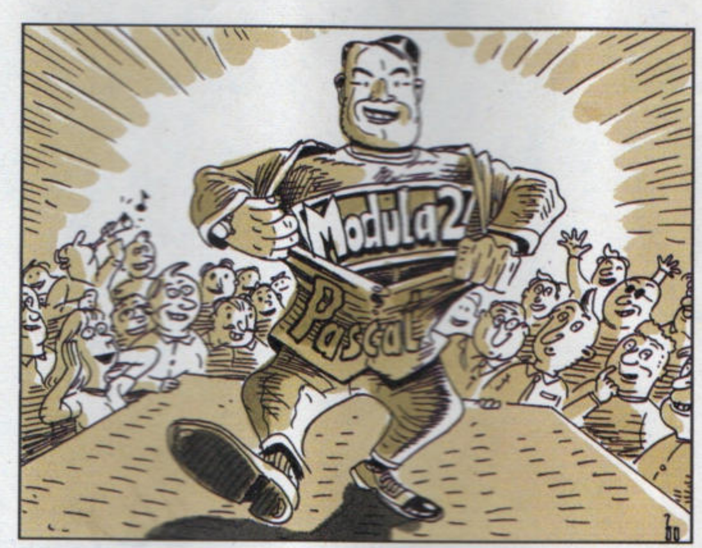
Und noch eine Sensation finden Sie auf der Leserdiskette zum Amiga: Als Automaten-Version begeisterte das fantastische Murmel-Spiel »Marble Madness« Tausende. Für die Amiga-Besitzer unter unseren Lesern haben wir deshalb den Level 1 dieses Super-Spiels auf die Leserservice-Diskette draufgepackt.

(Horst Brandl, Redakteur)

Inhalt



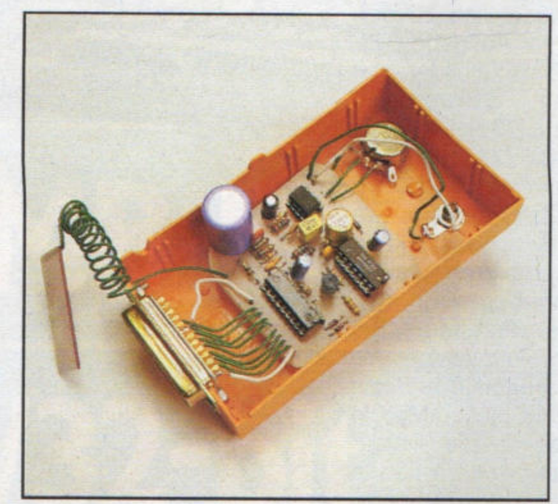
14 Der Trend zu größerer Leistung bei den Computern geht weiter. Apple hält mit. Der neue Macintosh Plus trägt mit 1-MByte-RAM-Speicher, dem 3½-Zoll-Diskettenlaufwerk und viel Komfort diesem Trend Rechnung.



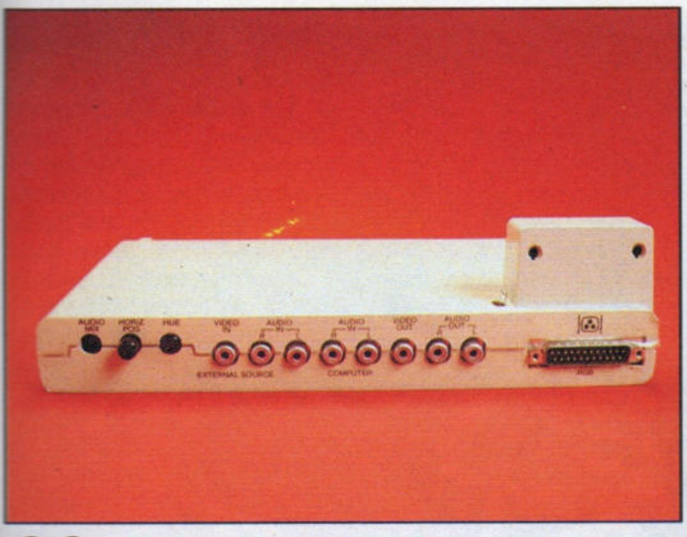
82 Modula hat sich zu einer der gefragtesten Programmiersprachen für den Atari ST gemausert. In unserem Kurs für Ein- und Umsteiger zeigen wir Ihnen, was diese Sprache zu einem so faszinierenden Werkzeug macht.



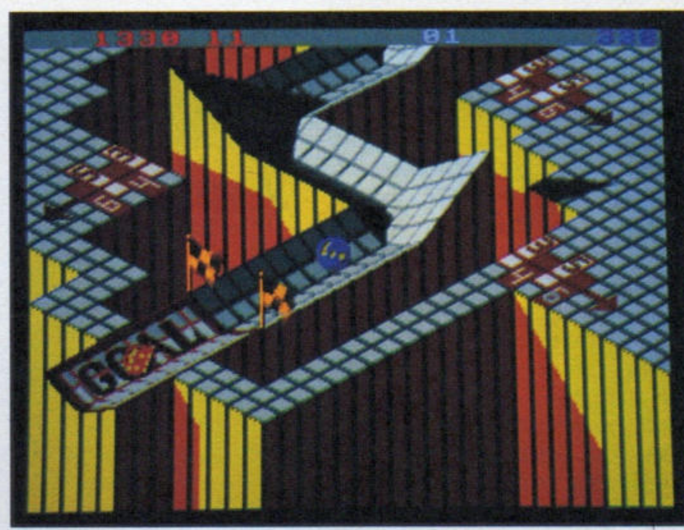
26
Lassen Sie sich von den fantastischen Bildern, die Sie mit einem Video-Digitizer auf Ihren Computer zaubern, in ein Wunderland entführen. Farben und Formen unterwerfen sich Ihrem Willen. Erschaffen Sie sich eigene Fantasiewelten.



40
Faszinieren den einen Farben, so versinkt der andere in Klängen. Ein Bastelkurs bietet den preiswerten Einstieg in die Klangdigitalisierung. Von der Hardware bis hin zur richtigen Software wird alles ausführlich erläutert.



32 Genlock, ein Zauberwort für Videobegeisterte. Was bisher Profis in den Fernsehstudios vorbehalten war, bringt ein passendes Interface Amiga-Besitzern nun ins Wohnzimmer. Der Videovorspann für jedermann.



36 »Marble Madness« heißt die Top-Umsetzung eines Automatenhits für den Amiga. Dieses und einige andere Superspiele für die 68000-Computer, sowie ein Spiele-Listing, stellen wir Ihnen vor.

Atari ST als Tonstudio

Sound-Digitizer selbst gebaut

Spitzen-Software im Test

für Atari St und Amiga

Kurse

Grundlagen

Anwendungs-Listings

Tips-und-Tricks-Listings

Vermischtes



USA – Land der Trends

Freak-Computer Apple II, Jedermann-Computer C64 und Prestige-Computer Macintosh – sie alle entstanden in den Vereinigten Staaten. Aber dort ruht man sich nicht auf den Lorbeeren aus. Hier nun, aktuelle Reiseeindrücke aus den USA zu neuen Trends im Computermarkt.

Zwei Besuchsziele stehen auf unserem Reiseplan: Chicago, im Nordosten der Staaten am riesigen Lake Michigan gelegen, und nach New York, die größte Wirtschaftsmetropole in den USA. Anschließend geht es in den Südwesten nach Kalifornien. Dort wollen wir Atari besuchen.

In Chicago erwartet uns hochsommerliche Hitze, die allerdings durch die ständige Brise vom Michigansee her gemildert wird. Kein Wunder, daß diese Sechs-Millionen-Metropole den Beinamen »windy city« trägt.

Hier findet alljährlich die Consumer Electronics Show (CES) statt. Heuer präsentierten 1400 Aussteller aus aller Herren Länder Produkte der Unterhaltungselektronik. Dazu gehören auch Heimcomputer. Auf dieser Messe sah man Trendsetter-Produkte ganz unterschiedlicher Art. Aber jeder spürte: 1986 ist das Jahr der 68000-Computer. Atari ST, Amiga, Macintosh, sie bildeten ein starkes Trio, an dem sich viele Aussteller orientierten. Oft gehörte Begründung: »Der Kunde stellt eben immer höhere Ansprüche. Die neuen schnellen Prozessoren erfüllen diese Ansprüche.«

Besonders die grafisch hochwertigen Benutzeroberflächen beeindruckten die Besucher sichtlich. Clive Smith, Präsident von Commodore, sieht aber keinen Grund, die Domäne »Benutzeroberfläche« kampflos den 16-Bit-Maschinen zu überlassen. Er bemerkte in einem Interview dazu: »Grafik rückt immer mehr in den Blickpunkt. Wir haben darauf schnell reagiert. Der C64 bekam die grafische Benutzerschnittstelle GEOS. Dadurch gewinnen wir sicher viele neue Kunden, die sich mit der bisherigen Kommandoschnittstelle nicht anfreunden können.«

Die Hersteller gehen also neue Wege. Das war nicht immer so: Vergleicht man CP/M mit MS-DOS, dann ergeben sich Unterschiede bei den Befehlen, aber die Grundstruktur hat sich nicht geändert. Anders bei den neuen grafischen Benutzerschnittstellen.

Da wurde nicht nur Altbewährtes verbessert, sondern von Grund auf neu konzipiert. Clive Smith erläuterte: »Wir Hersteller müssen neue Wege gehen. Bisher haben wir die Computer einfach immer leistungsfähiger und preiswerter

gemacht. Was dabei auf der Strecke blieb, war die Bedienung. Mit dem Amiga bringen wir ein neues Konzept. Jeder kann sofort mit diesem Computer umgehen. Noch nie war die Handhabung eines so leistungsstarken Computers schneller zu erlernen.« Man kann es auch so umschreiben: Viele Anwender wollen auf immer schnellere und größere Computerzüge aufspringen. Nur war ihnen bisher das Trittbrett zu hoch. Einziger Rat der Hersteller: Springt höher!

Jetzt entscheiden sie sich für den anderen und effektiveren Weg. Sie hängen das Trittbrett tiefer und erleichtern dem Anwender den Einstieg. Dieser einschneidende Trend in der Computergeschichte hat mit dem Macintosh begonnen und gewinnt rasch immer größere Dimensionen.

Noch weitere richtungweisende Entwicklungen wurden sichtbar. Die Softwarepreise tendieren stark nach unten. Gerade bei Atari ST und Amiga. Für ein Malprogramm in der Klasse von »Deluxe Paint« hätte man vor einigen Jahren einige tausend Mark auf den Ladentisch legen müssen. Heute zahlt man weniger als 300 Mark dafür. Und das ist kein Einzelfall. »Degas« von Batteries Included zum Beispiel avancierte schnell zum Bestseller für den Atari ST. Preis: 148 Mark.

Vor wenigen Jahren war das undenkbar. Batteries Included ist aber auch Verfechter einer weiteren Entwicklung: der Software ohne Kopierschutz. Michael Reichmann, Präsident von Batteries Included, erklärte dazu: »Degas wurde schnell zum Bestseller für den ST. Wir haben das Programm bisher 25000mal verkauft. Obwohl kein Kopierschutz auf der Diskette ist. Ein gutes Produkt und ein ausführliches Handbuch verkaufen sich trotz Raubkopierern. Das beweist Degas ganz eindeutig.«

Nächste Station auf unserem Trip durch die Vereinigten Staaten sollte das Sonnenland Kalifornien werden. Davon durften wir uns besonders aktuelle Informationen erwarten. Immerhin ist hier in und um das berühmte Computer-Mekka Silicon Valley herum alles versammelt, was in der Branche Rang und Namen hat. Und richtig: Bei Atari kristallisierten sich weitere Trends heraus.

Ataris Zentrale liegt in Sunnyvale, in einem neuen Industriegebiet an der Borregas Avenue. Ein flacher Bau, der sich im oberen Teil kelchartig verbreitert, ist das Hauptquartier »Atari-Lab«. Durch die ungewöhnliche Bauweise versucht man die Sonne auszusperren, von der man hier eher zuviel hat.

Glückliches Kalifornien! Aber auch wir hatten Glück. Eine persönliche Unterredung mit dem Chefentwickler von Atari, Shiraz Shivji, erwartete uns. Was viele nicht wissen: Ihm verdankt auch der Commodore 64 seine Existenz. Erst vor einigen Jahren wechselte Shivji zu Atari und entwickelte hier den 16-Bit-Preisbrecher Atari ST, der nach seiner Aussage erst der Anfang einer ganzen Computerreihe mit 16-Bit-Technologie ist.

Shivjis Büro, in das wir geführt werden, ist klein und mit erstaunlich wenig Technik ausgestattet. Lediglich ein Terminal für eine der drei VAX 11/780 zeugt von Computerpower. Die VAXen unterstützen ihn und seine Mitarbeiter bei der Entwicklungsarbeit. Eine davon simuliert seit Oktober 85 den Blitter. Erst wenn die Schaltung eines solchen Chips in der Simulation einwandfrei läuft, folgen die »echten« Testversionen.

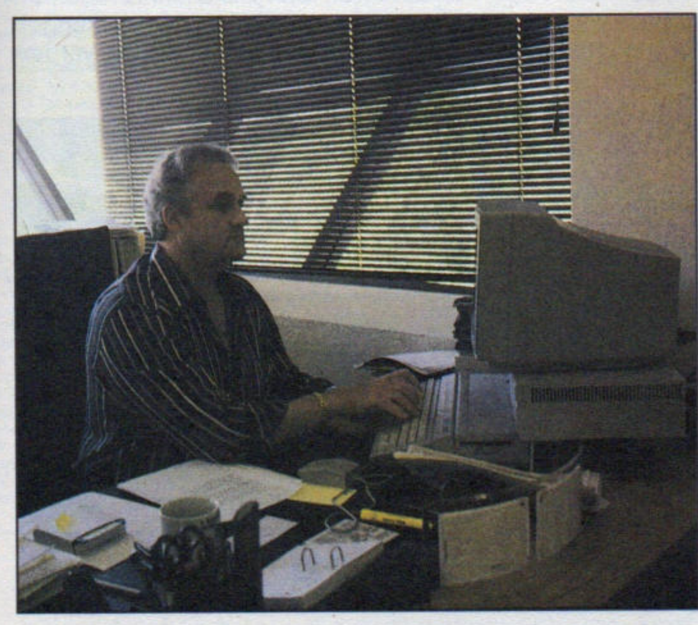
Erste Frage an den Atari-Manager: Woran arbeitet man gerade in der Computer-Schmiede? »Wir arbeiten an vielen neuen Sachen. Keine Firma kann sich auf ihren Lorbeeren ausruhen. Wir haben gerade erst begonnen, eine neue Computergeneration zu entwickeln. Der ST ist nur der Anfang einer Reihe weiterer Computer.«

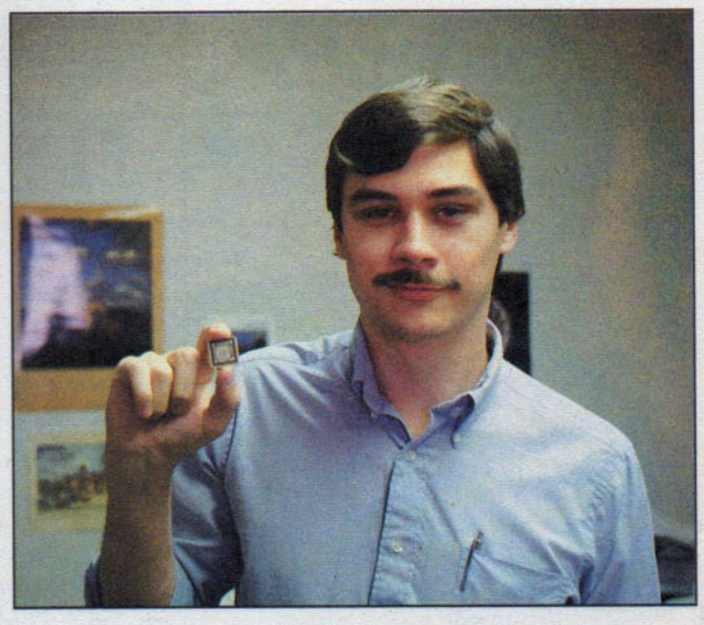
Wir melden Zweifel an der realen Existenz des Grafikprozessors »Blitter« an. Unsere Skepsis erfüllt ihren Zweck. Sie lockt ihn aus der Reserve. Er führt uns in das angrenzende Büro und stellt uns Duck Renn vor, den Chip-Spezialisten von Atari. Von ihm stammen »Glue« und »Shifter«.

Sie haben sich bereits in über 200 000 verkauften STs bewährt. Renn präsentiert uns stolz ein Testmuster des Blitter. Das gutgehütete Layout dieses schnellen Grafikprozessors hängt an der Wand.

Ihm sieht man nicht an, daß dieses Gewirr aus mehr als 20 000 Transistoren, das eher dem Stadtplan von Los Angeles gleicht, in Wirklichkeit nur rund 5 Quadratzentimeter groß ist. Wer findet sich da noch durch? Duck Renn! Er kennt jedes Detail seines Zöglings auswendig. Bis zu 20mal schneller wird mit diesem mikroskopischen Wunder die Grafik des ohnehin schon grafisch hochtalentierten Atari ST.

Das Prinzip ist einfach. So ein Grafikprozessor verfügt über eine Vielzahl


Sig Hartmann, Software-Chef von Atari, begutachtet Programme


Duck Renn, Chip-Spezialist, mit einem neuen Grafik-Prozessor

von Befehlen für Bit-Ansteuerung. Darin unterscheidet er sich von einer CPU, wie der MC 68000. Auch sie ist sehr schnell, aber ihr Befehlssatz ist Byte-orientiert. Möchte man zum Beispiel einen Speicherbereich Bit für Bit beeinflussen, dann erfordert das wesentlich mehr Aufwand. Auf dem Bildschirm aber ist ein Punkt eben nur ein Bit. Daher die geniale Begabung des Blitter für Grafik.

Ohne ihn müßte der Atari ST mit verbesserter Grafik auch teurer werden, und das würde der Marktstrategie des obersten Bosses, Jack Tramiel, zuwiderlaufen.

Auch bei der Farbpalette möchte Technikpapst Shivji höchsten Ansprüchen gerecht werden. Eine neue Grafikkarte steht kurz vor der Fertigstellung. Hinter einer schwer gesicherten Stahltüre, im Allerheiligsten der Entwicklungsabteilung, bekommen wir dann die wirklich geheimen Projekte zu sehen. Nur Shivji selbst hat die Berechtigung, uns hierher zu führen.

Er zeigt auf einen Probeaufbau mit zwei großen Platinen. Die erste stammt aus einem Atari ST, soviel ist sicher. Interessanter ist allerdings die zweite. Dieses Gewimmel aus kleinen ICs stellt den Testaufbau der neuen hochauflösenden Farbgrafikkarte dar. Ein Monitor ist angeschlossen und zeigt die fantastische Auflösung: 640 mal 480 Punkte in 16 Farben. In zwei Farben sogar unglaubliche 1280 mal 960 Punkte! Wichtiges Zentrum der Grafikkarte ist der Blitter. Man braucht ihn, um die dabei anfallenden großen Datenmengen schnell zu bewegen. Rasend schnell huschen Kreise, Rechtecke und Linien über den Bildschirm.

Aber auch die Hauptplatine spiegelt in der verbesserten Neufassung einen deutlichen Trend wider: Trotz höherer Leistung verringert sich die Zahl der Bauteile immer mehr. Statt sechs ROM-Bausteinen werden es in Zukunft nur noch zwei sein, denn Atari baut auf neuentwickelte 1-MBit-ROMs. Die hohe Auflösung allerdings fordert eine Menge Speicherplatz. Shivji plant ST-Versionen mit 2 bis 4 MByte RAM.

Testversionen mit 4 MByte laufen bereits. Computer mit soviel Speicherplatz und einer CPU aus der 68000er-Familie sind prädestiniert für Multitasking. Und da Motorola bei den Speicherverwaltungsbausteinen ohnehin Neuerungen angekündigt hat, will Shivji diese sofort bei seiner Entwicklung berücksichtigen. Die »Page Memory Management Unit« basiert auf der bekannten MMU im Atari ST. Sie ist für virtuelle Speicherverwaltung von Multitasking-Betriebssystemen ausgelegt.

Aber auch die CPU wird einer neueren Version weichen, der MC 68020. Die 68020 arbeitet mit echten 32 Bit, gegenüber der 68000, die zwar intern mit 32 Bit operiert, aber extern nur einen 16-Bit-Datenbus nutzt. Die Taktfreqenz steigt gleichzeitig von 8 auf 12,5 MHz. In seinem Inneren birgt die 68020 einen 256 Byte umfassenden Cache-Speicher. Bei jedem Speicherzugriff lädt der Prozessor immer 256 Byte »auf Vorrat« in seinen »RAM«. Durch diesen großen Speicher in seinem Inneren sinkt die Zahl der Zugriffe auf externe Bausteine. Der Geschwindigkeitsgewinn ist enorm. Um bei mathematischen Berechnungen die Ablaufgeschwindigkeit noch weiter in die Höhe zu treiben, setzt Shivji den Arithmetikprozessor 68881 ein.

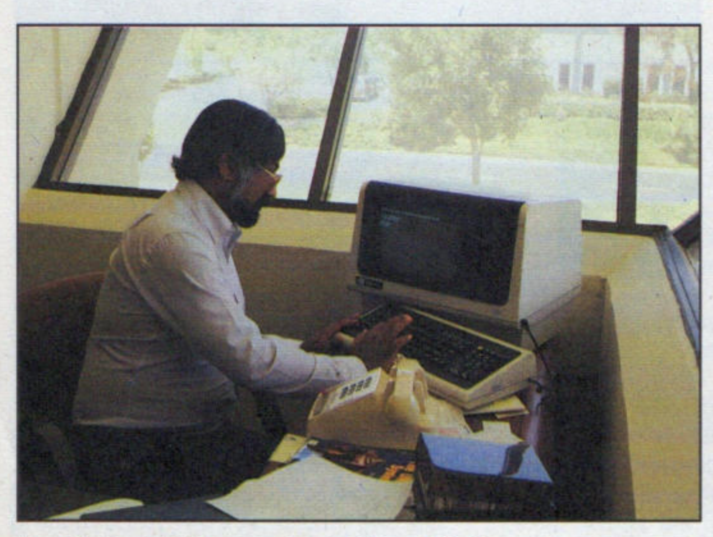
Mit einem geplanten Preis von umgerechnet 3500 Mark für den Endanwender bereitet Jack Tramiel eine weitere Preisbombe für den heißumkämpften Computermarkt vor.

Aber diese Zukunftsmusik ist für Shivji schon fast wieder Schnee von gestern. Er brütet bereits den Super-Computer von übermorgen aus! Um über Motorolas neueste Entwicklungen auf dem laufenden zu sein, steht er in engem Kontakt mit deren Entwicklern. Und Motorola hat Neues zu bieten, die CPU MC 68030! Shivji verriet uns: »Die 68030 ist eine Kombination aus 68020 und einem Teil der PMMU. Mit 20 MHz Taktfrequenz und 32-Bit-Architektur läuft sie fünfmal so schnell wie eine 68020-CPU.« Haupttrend also immer noch: Noch schneller, noch kleiner, noch höher integriert und dadurch noch leistungsfähiger.

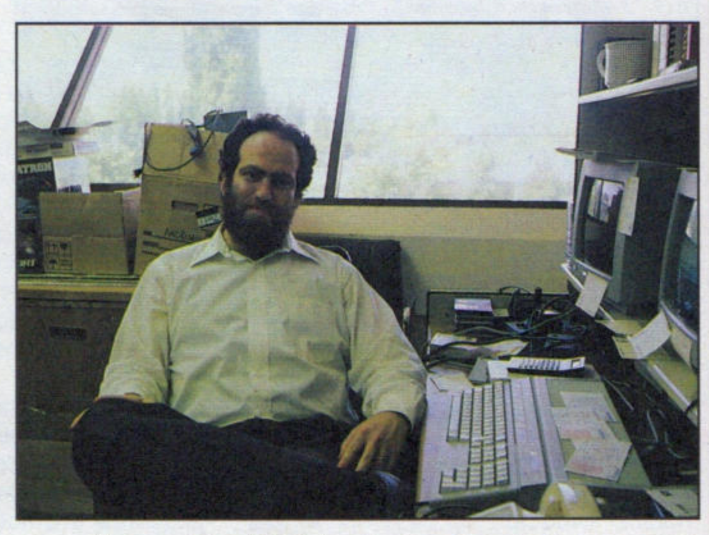
Shivji sieht die Zukunft für Atari aber auch bei der Peripherie. Er ist der Meinung, daß künftig grafische Anwendungen eine zunehmend größere Rolle spielen.

Nur Laserdrucker können die immer höher auflösenden Grafiken in hervorragender Qualität aufs Papier bringen. Atari möchte auch in diesem Markt alle Preisbarrieren einreißen. Ein Set aus Atari ST mit 2 MByte RAM, Monitor, Diskettenlaufwerk und Laserdrucker für unter 6000 Mark, so wird eine der Atari-Sensationen in diesem Bereich aussehen. Wie dieser Preis überhaupt möglich ist, erläutert Shivji mit wenigen Worten. 2 MByte RAM genügen seinen Worten nach, um einen Laserdrucker direkt anzusteuern. Mit Hilfe ausgeklügelter Software entfällt deshalb der teure Controller zwischen Computer und Drucker.

Denkt er, Shivji, manchmal auch an die gesellschaftlichen Auswirkungen seiner Arbeit? Sieht er neben vorder-



Der Chefentwickler von Atari, Shiraz Shivji...



...und Leonard Tramiel, Sohn des »großen« Jack



PROGRAMM-SERVICE



Bestellungen in der Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. 042/41 56 56
Bestellungen in Österreich: Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Tel. 0222/83 31 96,
Microcomput-ique E. Schiller, Fasangasse 21, A-1030 Wien, Tel. 0222/78 56 61,
Ueberreuter Media Handels- und Verlagsgesellschaft mbH, Alser Straße 24, A-1091 Wien, Tel. 0222/48 15 38-0
Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Das Angebot dieser Ausgabe:

Programme für die Atari ST

Disk1: Die komplette Software für den Sound-Digitizer: Quelltext – Demoprogramm – Sounddemos für alle STs
1 Diskette, Bestell-Nr. LH 86S9 D1 **DM 29,90*/sFr. 24,90/öS 299,-***

Disk2: C-Library für Spiele – Druckeraccessory – Bildschirmeffekte – Filekill – Fastloader für ROMs – GEM-Menü unter Basic – schneller Spriteeditor – GOLEM-Texteditor – Druckercodewandler – ST-Zeichensatz für Epson und Kompatible – Prozessorstatus-nie mehr Bömchen
1 Diskette, Bestell-Nr. LH 86S9 D2 **DM 29,90*/sFr. 24,90/öS 299,-***

Disk3: Bibliotheken für Fortran und Modula – GOLEM-Texteditor – Superfilter – GEM-Bagger
1 Diskette, Bestell-Nr. LH 86S9 D3 **DM 29,90*/sFr. 24,90/öS 299,-***

Alle Disketten im Paket
3 Disketten, Bestell-Nr. LH 86S9 D4 **DM 69,90*/sFr. 59,90/öS 699,-***

Programme für Commodore AMIGA
Marble Madness Demo – Maze – Animationsprogramme
1 Diskette, Bestell-Nr. LH 86S9 D5 **DM 29,90*/sFr. 24,90/öS 299,-***

Programme aus früheren Ausgaben:

Happy-Computer, Ausgabe 9/86
Schneider-Computer
Aus Ausgabe 8/86
Angriff der Cyclonen: Seit Jahrhunderten liegen die Menschen und die Cyclonen miteinander im Kampf. Sie dringen in die Tiefen unserer Galaxie vor und wehren den neuesten Angriff auf unseren Heimatplaneten ab.
Soundeditor: Neben hervorragender Grafik besitzt der Schneider auch einen hervorragenden Tongenerator. Menügesteuert ist dieser ganz einfach zu bedienen.
Aus Ausgabe 9/86
Exdisc: 13,5 KByte mehr Speicherplatz auf jeder 3-Zoll-Diskette. Kitzeln Sie das letzte freie Bit aus Ihrem Massenspeicher.
Felix und der Maulwurf: Retten Sie Ihren Garten vor dem kleinen Unhold. Ein Spiel voller Action.
...und alle Tips&Tricks aus den letzten beiden Ausgaben von Happy-Computer.
Bestell-Nr. LH 8609 SD (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 8609 SK (Kassette)
DM 34,90*/sFr. 29,50/öS 349,-*

Happy-Computer, Ausgabe 8/86
Commodore 64, Commodore 128
Bundesligamanager: Versuchen Sie Ihr Glück als Manager eines Bundesligavereins. Werden Sie Deutscher Meister, oder bewähren Sie sich im Kampf gegen den Abstieg.
Ultraboot Menue: Laden Sie Ihre Programme mit einem Tastendruck von Diskette. Lademenü für »Ultraboot«.
Earthraid: Listing des Monats. Verteidigen Sie die Erde gegen den Angriff gefährlicher Killermollusken.
Let's Bounce: Listing des Monats. Steuern Sie Ihren Tennisball sicher über die Hochhäuser. Vermeiden Sie dunkle Abgründe und gefährliche Bergspitzen.
Bestell-Nr. LH 8608 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 7/86
Schneider-Computer
Grafik-Gigant inkognito: Sensationell: 640 x 400 Punkte Auflösung für den CPC 464!
Explora 1.0: Eingabehilfe (Prüfsummer) für sämtliche Basic-Programme.
Grafikbär: Grafikbildschirme platz- und zeitsparend gespeichert.
Spritzige Sprites: Spritegenerator unter Nutzung der Befehlserweiterung »Toolbasic 1.0«.
Zeichen-Designer: Komfortabler und leistungsfähiger Zeichensatz-Generator.
Windows im ST-Look: Extrem schnelle Pull-Down-Menüs durch neue RSX-Window-Befehle.
Preiswerte Sicherheit: Kopiert vollautomatisch Programm-Dateien von Diskette auf Kassette.
Schwarz auf weiß: Endlich eine Hardcopy-Routine für alle drei Schneider-CPCs.
Disketten-Menü für dBase II: Utility für erhöhten Bedienungskomfort.
Horrible Halls: Spiel des Monats mit fantastischer Grafik, Spritedesigner und Construction-Set.
RSX-Fill: Schnelle Füllroutine als RSX-Befehlserweiterung.
Schnelle Kreise: Eleganter und vor allem schneller Basic-Algorithmus für Kreise.
Bestell-Nr. LH 8607 SD (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*

Happy-Computer, Ausgabe 6/86
Commodore 64, Commodore 128
Bestell-Nr. LH 8606 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 5/86
Commodore 64, Commodore 128
Bestell-Nr. LH 8605 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 4/86
Schneider CPC
Best.-Nr. LH 8604 SK (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*
Best.-Nr. LH 8604 SD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 3/86
Commodore 64/Commodore 128
Bestell-Nr. LH 8603 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 2/86
Commodore 64
Bestell-Nr. LH 8602 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 1/86
Commodore 64/Commodore 128
Bestell-Nr. LH 8601 CD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 12/85
Atari 800XL/130XE/800
Bestell-Nr. LH 8512 B (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 12/85
Schneider CPC
Bestell-Nr. LH 8512 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*
Bestell-Nr. LH 8512 D (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*

Happy-Computer, Ausgabe 11/85
Commodore 64
Bestell-Nr. LH 8511 A
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 10/85
Sinclair Spectrum
Bestell-Nr. LH 8510 D
DM 19,90*/sFr. 17,-/öS 199,-*
Atari 800XL
Bestell-Nr. LH 8510 B (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 9/85
Commodore 64
Bestell-Nr. LH 8509 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 8/85
Schneider CPC 464
Bestell-Nr. LH 8508 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 7/85
Commodore 64
Bestell-Nr. LH 8507 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 6/85
Commodore 64
Bestell-Nr. LH 8506 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 5/85
Schneider CPC 464
Bestell-Nr. LH 8505 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 4/85
Commodore 64
Bestell-Nr. LH 8504 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Computer, Ausgabe 3/85
Schneider CPC 464
Bestell-Nr. LH 8503 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*

Happy-Sonderhefte

Sonderheft 8/86: Computer als Hobby
Bestell-Nr. LH 86S8 D1
DM 29,90*/sFr. 24,90/öS 299,-*
Bestell-Nr. LH 86S8 D2
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 86S8 D3
DM 29,90*/sFr. 24,90/öS 299,-*

Sonderheft 7/86: Schneider
Bestell-Nr. LH 86S7 SD (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 86S7 SK (Kassette)
DM 34,90*/sFr. 29,50/öS 349,-*

Sonderheft 6/86: 68000
Programme für Atari ST
Bestell-Nr. LH 86S6 D1
DM 34,90*/sFr. 29,50/öS 349,-*
Forth-Compiler für Atari ST
Bestell-Nr. LH 86S6 D2
DM 29,90*/sFr. 24,90/öS 299,-*
Programme für Apple Macintosh
Bestell-Nr. LH 86S6 D3
DM 34,90*/sFr. 29,50/öS 349,-*

Sonderheft 5/86: Programmiersprachen
Bestell-Nr. LH 86S5 SD, für Schneider
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 86S5 CD, für C64
DM 29,90*/sFr. 24,90/öS 299,-*
Bestell-Nr. LH 86S5 8D, für C128
DM 29,90*/sFr. 24,90/öS 299,-*

Sonderheft 4/86: Schneider
Bestell-Nr. LH 86S4 K (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*
Bestell-Nr. LH 86S4 D (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*

Sonderheft 3/86: 68000
Bestell-Nr. LH 86S3 D (Diskette)
DM 29,90*/sFr. 24,90/öS 299,-*

Sonderheft 2/86: ATARI
Bestell-Nr. LH 86S2 D (2 Disketten)
DM 34,90*/sFr. 29,50/öS 349,-*

Sonderheft 1/86: Schneider
Bestell-Nr. LH 86S1 D (Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 86S1 K (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*

Sonderheft 2/85: Schneider
Bestell-Nr. LH 85S2 D (3"-Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 85S2 V (5¼"-Diskette)
DM 34,90*/sFr. 29,50/öS 349,-*
Bestell-Nr. LH 85S2 K (Kassette)
DM 29,90*/sFr. 24,90/öS 299,-*

Sonderheft 1/85: Spectrum
Bestell-Nr. LH 85S1 D (Kassette)
DM 19,90*/sFr. 17,-/öS 199,-*

* inkl. MwSt. Unverbindliche Preisempfehlung

Bitte verwenden Sie für Ihre Bestellung und Überweisung die eingeheftete Postgiro-Zahlkarte,
oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung.
Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

gründiger Faszination auch Gefahren? Abwägende Antwort: Natürlich, seiner Meinung nach muß sich vor allem die Einstellung vieler Menschen zum Computer ändern. Der unbedarfte Anwender sieht immer noch in erster Linie die Maschine und nicht das Ergebnis.

Steigt man in einer Firma auf EDV um, höre man oft: Wir sind auf Computer umgestiegen. Und nicht: Wir schaffen unsere Arbeit schneller. Oder fragt man zum Beispiel einen jungen Computerfreak, wenn er mit dem Malprogramm »Degas« am Computer malt, was er tue, antworte dieser bestimmt: »Ich sitze am Computer« und nicht: »Ich male«. Für viele stehe der Computer im Vordergrund.

Der sei aber nur ein Werkzeug, wie ein Hammer, mit dem man einen Nagel in die Wand schlägt, um ein Bild aufzuhängen. Niemand, der das Bild jemandem zeige, würde sagen: »Toll, was? Ich habe mit einem Hammer den Nagel selbst in die Wand geschlagen«, sondern über das Bild reden. Shivji: »Wir müssen umdenken.«

Um zu hören, welche Trends es bei der Software gibt, sprachen wir mit Sig Hartmann, Software-Präsident von Atari. Große Absatzchancen wittert er bei Universitäten und Firmen. Für die dort eingesetzten Großrechner sind STs ideale »intelligente« Terminals. Die Vorteile liegen auf der Hand. Zum einen kostet ein herkömmliches Terminal selbst ohne Systemintelligenz wesentlich mehr als der ST. Zum anderen ist ein eigenständiges System wie der ST weit vielfältiger einsetzbar.

Die nötige Software bietet Atari bereits jetzt. Eine Vielzahl Terminalemulatoren findet man in dem umfangreichen Software-Katalog.

Trotzdem ist Sig Hartmann an weiteren Emulatoren interessiert.

Richard Frick, Direktor der ST-Produktlinie, unterstützt ihn dabei. Er adaptiert zur Zeit das Betriebssystem BOS an den ST. Eine ganze Reihe Betriebssysteme wurden auf den ST übertragen, zum Beispiel das Multiuser- und Multitasking-System OS9 (Einen umfassenden Bericht finden Sie in dieser Ausgabe).

Aber auch die Einsteiger werden von Atari nicht als Stiefkinder behandelt. Richard Frick und Leonard Tramiel, einer der Söhne des Firmenbosses Jack Tramiel, kümmern sich um eine verbesserte Version des ST-Basic. Durch viele Fehler in der alten Version konnte bisher beim Programmieren keine rechte Freude aufkommen. Beim Ausmerzen dieser Fehler geht man mit viel Sorgfalt zu Werke. Wie uns Leonard Tramiel versicherte findet aber nicht nur eine Fehlerberichtigung statt. Es werden auch weitere Befehle integriert. Bald huschen unter Atari-Basic sogar Sprites über den Bildschirm. Erstaunliche Erkenntnis laut Frick: Trotz C, Pascal und Modula-2 lebe auch auf den Computern der neuen Generation Basic weiter.

Apple läutete mit seinem Macintosh (genaugenommen mit der Vorläuferin »Lisa«) eine neue Ära, die der 68000-Computer, ein – aber nur wenige hörten das Signal. Atari erst lieferte mit seiner ST-Serie den Paukenschlag, der die Massen aufhorchen ließ. Amerika ist durch solche Firmen und Männer wie Shiraz Shivji auch heute noch das Geburtsland für neue Trends.

(hb)

Harte Scheiben für schnelle Daten

Massenweise Massenspeicher von Atari! Die langerwartete Festplattenstation SH204 macht den Atari ST zum zwanzigfachen »Plattenmillionär«.

Seit Beginn des Jahres 1985 wurde die bis dahin satt und träge vor sich hin dämmernde Heim- und Personal-Computer-Szene mehrfach gründlich durchgeschüttelt. Wähnte man sich vorher mit 128 KByte RAM im Computer und zweimal 400 KByte Kapazität auf den Diskettenstationen noch völlig ausreichend mit Speicherplatz versorgt, so erntete man plötzlich bei Neueinsteigern unter den Computeranwendern mit der teuer bezahlten Computeranlage bald nur noch ein leicht amüsiertes oder bestenfalls mitleidiges Lächeln.

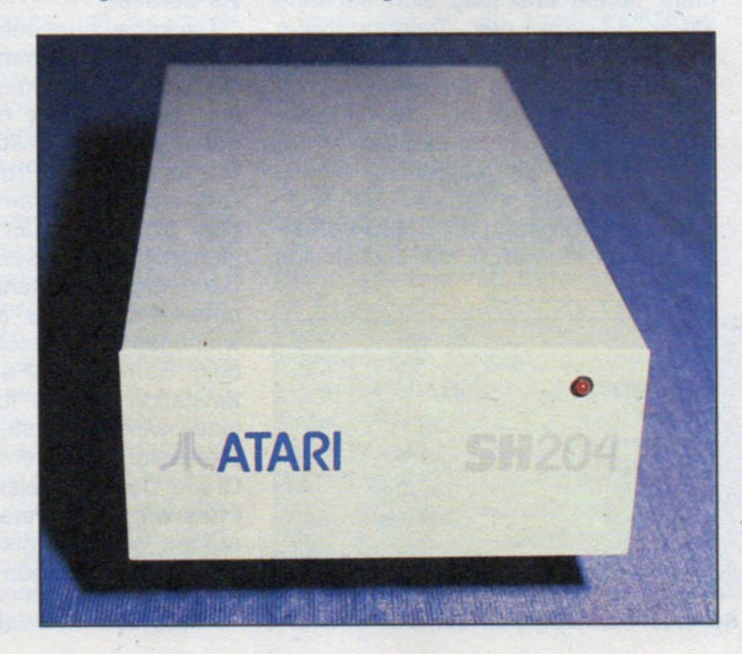
Die Zentralspeicher der Computer wuchsen zu Millionären heran, die Diskettenstationen wurden allmählich zu wirklichen Massenspeichern und eine bis dahin für nur wenige äußerst begüterte Computerbesitzer erschwingliche Datenspeichereinheit, die Festplattenstation, auf Computerdeutsch auch Harddisk genannt, rückte preislich in den Etatrahmen kleinerer Büros oder gar von Privathaushalten. Natürlich begünstigte das weltweite Sinken der Preise für Speicherchips diese Entwicklung. Es bedurfte aber dennoch des Wagemutes eines der großen Alten im Computergeschäft, um den wirklichen Giganten und die vielen Möchtegernriesen der Zunft vom Pfade der Hochpreispolitik und Innovationsmüdigkeit abzubringen.

Gemeint ist Jack Tramiel und die wiedergeborene Atari Corporation, die ohne Angst vor dem unkalkulierbaren Risiko ein wirklich neues Computersystem aus dem Boden stampften und zu Preisen auf den Markt brachten, die beste und fortschrittlichste Computertechnologie auch für den kleinen Geldbeutel erschwinglich machte.

Von Sektoren und Zylindern

Als neuestes Produkt im ST-System glänzt die schon im Herbst 1985 auf der Systems in München vorgestellte und seit langem heißerwartete Harddisk SH204. Für unter 2000 Mark ist damit eine Massenspeichereinheit verfügbar, die aufgrund ihrer Kapazität von 20 Megabyte und einer schnellen Datenübertragungsrate dem Atari ST weitere Anwendungsgebiete erschließt. Im unscheinbaren grauen Stahlblechgehäuse mit dem blauen Atari-Schriftzug steckte bei unserem Testexemplar ein vertrauenerweckend massiv aussehendes Laufwerk ST225 der Firma Seagate.

Netzteil, Lüfter und zwei Platinen vervollständigen den Inhalt des Gehäuses.



Datenmassen flott gebändigt: Speicherriese in schlichtem Atari-Grau

Auf der größeren Platine befindet sich der Harddisk-Controller von der Firma Adaptec, während die kleine Platine, ein Eigenprodukt von Atari, als sogenannter Hostadapter für die Anpassung der Signal- und Steuerleitungen an den DMA-Port des Computers verantwortlich zeichnet.

Der Speicherplatz auf Festplattenlaufwerken ist ähnlich wie auf Disketten in Spuren und Sektoren organisiert. Da aber Harddisks meist über mehr als eine Speicherplatte verfügen, spricht man noch von einer weiteren Organisationseinheit, dem sogenannten Zylinder. Zu einem Zylinder gehören alle Spuren, die auf den eingebauten Platten übereinanderliegen. Das Handbuch zur SH204 gibt die Anzahl der Spuren mit 2448 in 612 Zylindern an. Jede Spur ist in 17 Sektoren unterteilt. Daraus ergibt sich ein Aufbau des ST225-Laufwerks aus zwei beidseitig beschreibbaren Platten mit vier Schreib/Lese-Köpfen. Bei einer Sektorgröße von 512 Byte macht das rein rechnerisch eine Gesamtkapazität von 21307392 Byte. Die Datenübertragungsrate beträgt laut Hersteller 5 Megabit pro Sekunde.

Die mitgelieferte Software besteht aus der Treibersoftware ADHI.PRG und den zwei Dienstprogrammen HDX.PRG und SHIP.PRG. Der Harddisk-Treiber ADHI.PRG arbeitet nur auf ST-Computern mit ROM-Betriebssystem. Bei Arbeitsbeginn lädt man zuerst die Treibersoftware in den Zentralspeicher. Vor der ersten Benutzung formatiert man die SH204. Dazu dient das Programm HDX.PRG, das noch über weitere Funktionen verfügt. Beim Formatieren (Menüpunkt »Format«) werden die 20 Megabyte des physikalischen Laufwerkes in drei logische Laufwerke, die sogenannten Partitions eingeteilt, die wie drei getrennte Diskettenlaufwerke ansprechbar sind. HDX.PRG erzeugt die logischen Laufwerke C, D und E mit 4, 6 und 10 Megabyte Kapazität.

Der Befehl »Partition« im HDX-Programm legt diese Einteilung nach eigenen Wünschen fest. Es lassen sich bis zu vier logische Laufwerke installieren. Die Maximalgröße einer Partition beträgt 16 MByte.

Jede einzelne Partition ist über den Menüpunkt »Zero« unabhängig von den anderen Partitions löschbar. Die Funk-

tion »Markbad« sucht defekte Sektoren in den einzelnen Partitions und markiert sie. Sie gelten danach für das Betriebssystem als belegt und stehen zur Datenspeicherung nicht mehr zur Verfügung. Partition C besitzt eine bevorzugte Stellung im Betriebssystem. Beim Einschalten oder beim Betätigen der RESET-Taste sucht der ST zuerst hier nach Desktop-Accessories oder der Datei »DESKTOP.INF«.

Das dritte Programm auf der beigepackten Diskette benötigen Sie, wenn die SH204 transportiert werden soll. Die Schreib/Lese-Köpfe berühren die Speicherplatten der Harddisk während des Betriebs nämlich gar nicht, sie schweben auf einem Luftkissen Bruchteile von Millimetern über der Platte. Eine Berührung würde die empfindliche Plattenbeschichtung zerstören und damit gleichzeitig die an der Berührungsstelle gespeicherten Daten. Daher bringt man vor Transport der SH204 die Schreib/Lese-Köpfe in eine »Parkposition«, an der keine Daten gespeichert sind. Genau diese Aufgabe erfüllt »SHIP.PRG«.

Die Platten der Harddisk laufen nach Abschalten des Laufwerkes mehrere Minuten nach. Rütteln oder Verkanten des Laufwerkes während der Nachlaufzeit kann zur Zerstörung der Platten oder der Köpfe führen. Atari rät zu einer Wartezeit von etwa zehn Minuten zwischen Abschalten und Transport der SH204.

Verwaltung kein Problem

Nach soviel Vorbereitung aber nun endlich zur eigentlichen Arbeit mit Ataris neuem Prunkstück! Beginnen wir mit etwas Kritik. Die gewohnte Stille am heimischen Schreibtisch ist dahin. Die SH204 läßt den Benutzer über ihren Arbeitseifer nicht im Unklaren. Das Betriebsgeräusch ist in ruhigen kleinen Arbeitsräumen unüberhörbar.

Allerdings gewöhnt man sich recht schnell an das rauschende Summen der Festplattenstation. In größeren Büros dürfte der höhere Grundgeräuschpegel dieses Summen ohnedies verdecken. Die Vorteile der Computerarbeit mit der Harddisk lassen diesen kleinen Nachteil jedoch schnell in Vergessenheit geraten. Dabei liegt der wichtigste Fortschritt gar nicht einmal in der enormen Speicherkapazität. Viel deutlicher wird der Unterschied in der Datenübertragungsgeschwindigkeit beim Speichern und Laden. Einige Beispiele sind in der Tabelle aufgeführt Auch Compilerläufe beim Programmieren erinnern fast schon an die Geschwindigkeit einer RAM-Disk. Allerdings darf man bei Verwendung der Harddisk, nach einer nächtlichen Programmiersitzung von Müdigkeit übermannt, seinen ST getrost ausschalten, ohne das revolutionäre Produkt der harten Nachtarbeit in das Nirwana stromloser RAMs zu schicken.

Die vielen Gerüchte über die Probleme mit der Atari-Harddisk, die die glücklichen Besitzer der vereinzelten Prototypen angeblich an den Rand des Wahnsinns getrieben haben, konnte der bisherige Testbetrieb nicht bestätigen.

Weder Datenverlust bei Belegung mit mehr als 4 Megabyte noch Schwierigkeiten bei der Verwaltung von mehr als 40 Ordnern wurden bisher beobachtet (unser Testlaufwerk mit drei Partitions ist augenblicklich bereits zur Hälfte gefüllt und enthält 51 Ordner). Im amerikanischen Handbuch zur SH204 findet man allerdings den Hinweis, daß der ST auf allen angeschlossenen Massenspeichereinheiten insgesamt nicht mehr als vierzig Inhaltsverzeichnisse einschließlich der Ordner verwalten kann. Diese Angabe ist sicherlich nicht mehr richtig und mag sich auf eine ältere Version der Treibersoftware beziehen. Leider konnte uns Atari Deutschland für die ausgelieferte Version keine Angaben über einen Höchstwert der Ordneranzahl und die Art der Probleme bei Überschreitung dieses Wertes machen.

Bei der Erprobung der SH204 offenbarte sich eigentlich nur ein wirklich gravierendes Ärgernis. Dieses ist jedoch in keiner Weise der Harddisk anzulasten. Viele Programmierer und Softwareproduzenten berücksichtigten offensichtlich nicht, daß Arbeiten mit dem Atari ST erst mit einer Festplattenstation professionelle Dimensionen erreicht. Das sicherlich berechtigte Interesse, Software vor unerlaubtem Kopieren zu schützen, hat Schutzmechanismen hervorgebracht, die eine Benutzung der Harddisk geradezu unmöglich machen, und zwar auf Dauer. Nicht von Dauer dagegen ist die Wirksamkeit des Kopierschutzes. Die Kopierprogramme könnten noch stets mit der Entwicklung der Schutzmechanismen mithalten. Die ST-Software macht hier sicherlich keine Ausnahme. Noch ärgerlicher ist aber die Tatsache, daß einige Programme die Möglichkeiten des Betriebssystems, besonders hinsichtlich der Benutzung von Ordnern, nicht richtig nutzen.

Da greifen Programme beim Laden von RSC-Dateien auf ein fest einprogrammiertes Laufwerk zu, statt (was programmtechnisch ohne weiteres möglich wäre) Startlaufwerk und Startordner abzufragen und beim Nachladen zu berücksichtigen. Da sind andere Erzeugnisse der Programmierkunst in Ordnern gar nicht lauffähig, oder aber das Einlesen von Texten oder Daten muß jedesmal durch wilde »Klick«-Orgien in der Dateiauswahlbox auf das richtige Laufwerk oder auf den richtigen Ordner gelenkt werden, weil das Programm konsequent auf Laufwerk A als Datenlaufwerk beharrt. Dies ist nur ein kleiner Ausschnitt aus den bösen Überraschungen, die so manches an sich gute Programm dem Harddisk-Benutzer bereitet.

Dankenswerterweise sind jedoch auch viele Programme auf dem Markt, die, beabsichtigt oder nicht, vorzüglich mit der Harddisk harmonieren. Der geringe und nach Atari-Manier sicherlich noch fallende Preis der Festplattenlaufwerke läßt derartige Massenspeicher sicherlich zum Standard für leistungsfähige ST-Systeme aufsteigen. Dann wird die Lauffähigkeit der angebotenen Programme auf der Harddisk wichtiges Entscheidungskriterium beim Softwarekauf sein. Die Festplattenstation SH204 führt den Atari ST einen weiteren Schritt in die Welt der professionellen Computeranwendung. Große Computerleistung zu geringem Preis war 1985 versprochen worden. Auf ein Wort des großen Jack scheint man sich immer noch verlassen zu können!

(W. Fastenrath/hb)

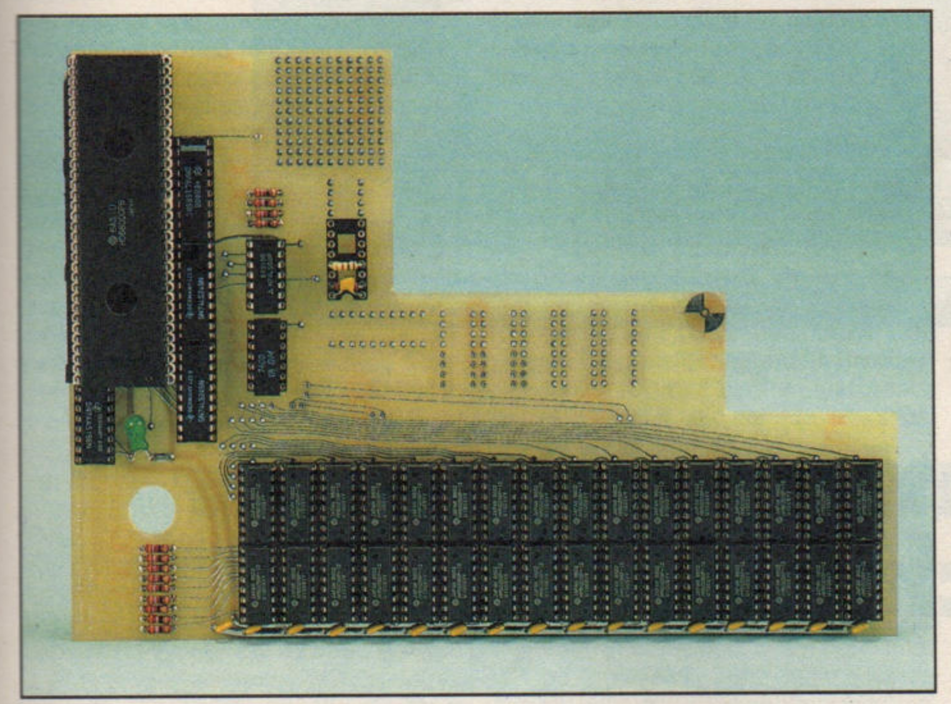
Geschwindigkeitstest

Atari Hard Disk SH204	Hard Disk SH204	Floppy Disk SF314
Ordner von Partition D nach Partition E kopieren (56 Dateien/764 KByte)	3,0 Minuten	-
Denselben Ordner löschen	22,0 Sekunden	-
ST-BASIC laden (144 KByte)	4,7 Sekunden	21,4 Sekunden
1st_WORD PLUS laden (153 KByte)	7,8 Sekunden	27,3 Sekunden
Text (22 KBytes) laden innerhalb 1st_WORD PLUS	9,8 Sekunden	15,5 Sekunden

Datentransfer mit harter und mit weicher Scheibe: Profitempo für den ST

Millionen für den Amiga

Mit Speicher ist der Amiga bekanntlich nicht gerade reichlich gesegnet. Die Speichererweiterung »DRAM-EX 4M« vergrößert den Speicher intern um 1 bis 4 MByte und bietet zusätzlich noch einige interessante Ausbaumöglichkeiten für die Zukunft.



Die RAM-Erweiterung wird fest in den Amiga eingebaut

In der Entwicklungsphase des Amiga waren RAM-Bausteine noch sehr teuer. Nur so erklärt es sich, weshalb ein Computer mit derartigen Grafik-, Sound- und Multitasking-Fähigkeiten nur über bescheidene 256 KByte-RAM-Speicher verfügt. Damit sind jedoch die wenigsten Programme lauffähig, sinnvolle Anwendungen ergeben sich erst ab 512 KByte. Doch auch diese Grenze ist recht knapp bemessen, und man sucht nach weiteren Aufrüstungen, um beispielsweise die RAM-Disk verstärkt einzusetzen.

»DRAM-EX 4M« nennt sich eine Erweiterung, die den RAM-Speicher des Amiga um 1 bis 4 MByte vergrößert. Die RAM-Karte wird in den Amiga eingebaut und hält so den Expansionsport für andere Module oder Geräte frei. Der Umbau geht, übrigens ganz ohne Lötkolben, so vonstatten: Computer aufschrauben, Abschirmblech entfernen, 68000-Prozessor heraushebeln, RAM-Karte einstecken, Prozessor daraufstecken und alles wieder zuschrauben – schon ist man um eine Million freie Speicherstellen reicher.

Doch halt – die hardwaremäßige Umrüstung erfordert auch eine kleine Software-Änderung. Fast alle Amiga-Disketten sind mit einer Autostart-Sequenz versehen, die bestimmt, welche Befehle nach dem Starten der Diskette auszuführen sind. In diese Befehlsliste reiht man einfach die Anweisung »ADDMEM« ein, um dem Betriebssystem mitzuteilen, daß mehr als 512 KByte Speicher zur Verfügung stehen.

Am besten arbeitet die RAM-Erweiterung mit der neuen Betriebssystem-Version 1.2 zusammen, da diese Workbench ein Symbol für eine RAM-Disk enthält. Diese RAM-Disk verhält sich bis auf zwei Unterschiede wie eine richtige Diskettenstation. Beim Ausschalten des Computers oder bei einem Reset gehen – im Gegensatz zur echten Diskette – alle Daten verloren.

Die RAM-Disk hebt sich andererseits durch sehr hohe Geschwindigkeit hervor und belegt nur genau soviel Speicher, wie ihre Programme benötigen. Wer mit Kickstart/Workbench 1.1 arbeitet, kann die RAM-Disk leider nur vom CLI aus ansprechen.

Mitten im Programm spielt die Betriebssystem-Version keine Rolle. Alle Dateien mit dem Zusatz »RAM:« vor dem Filenamen werden in jedem Fall von der RAM-Disk gelesen oder darauf gespeichert. »Deluxe Paint« erlaubt beispielsweise, eine ganze Bilderdiskette in die RAM-Disk zu kopieren und von dort aus jedes Bild schnell abzurufen. Bereits in der 1-MByte-Version der DRAM-Karte findet eine randvolle Diskette (880 KByte) ohne Schwierigkeiten Platz.

Die Vorteile einer RAM-Erweiterung sind vielfältig. Auch längere Programme arbeiten jetzt im Multitasking-Betrieb mit anderer Software zusammen, und Textverarbeitungen bieten einen riesigen Textspeicher. Wer selbst in Assembler oder C programmiert, wird nie wieder ein Programm ohne RAM-Disk assemblieren oder compilieren wollen.

In der Grundausstattung verfügt die »DRAM-EX 4M«-Karte über 1 MByte Speicher. Der Austausch der 256-KBit-Chips gegen 1-MBit-RAM-Bausteine rüstet die Karte auf 4 MByte auf. Leider sind diese Mega-Chips zur Zeit noch sehr teuer, so daß man sich vorerst wohl mit der 1-MByte-Version zufrieden gibt.

In Kürze erhalten Sie für die Karte auch eine batteriegepufferte Echtzeituhr, die den Amiga immer mit der richtigen Zeit und dem richtigen Datum versorgt, auch wenn das Gerät nicht läuft. Die Bohrungen für die entsprechenden Bausteine sind bereits auf der Platine vorhanden.

Ebenfalls schon vorbereitet ist die Karte für die Aufnahme des Arithmetik-Coprozessors NEC 77230, der den Amiga bei Fließkomma-Berechnungen bis zu 50mal schneller macht und kompatibel zu bestehender Software ist! Hier einige Daten dieses Mathe-Prozessors, bei denen man ins Schwärmen gerät: 1024 Register mit 32 Bit Breite, acht davon sind sogar 55 Bit breit; Rechengeschwindigkeit: 6,5 MFLOPs (Fließkommaberechnungen pro Sekunde in Millionen).

Die »DRAM-EX 4M«-Speichererweiterung überzeugt durch die einfache Installation und die interessanten Erweiterungsfähigkeiten, die den stolzen Preis von 1173 Mark rechtfertigen.

(ts)

Info: Alphatron Computersysteme, Tel.: 09131-250-18





Macintosh Plus, der Schritt weiter

Professionelle Computer-Anwendungen und mächtige Programme verlangen einen großen RAM-Speicher und leistungsfähige Laufwerke. Der neue Macintosh Plus bietet neben dem 1-MByte-Arbeitsspeicher noch hierarchische Dateiverwaltung, direkte Anschlüsse für Festplatte beziehungsweise Streamer und eine integrierte Tastatur.

Wer sich unter Mac-Benutzern umhört, gewinnt schnell ein einheitliches Bild von deren Wunschliste: »Für den professionellen Einsatz muß der Mac schneller werden, der Diskettenzugriff dauert einfach zu lange«, war da immer wieder zu hören. Auch der umständliche Wechsel zwischen zwei Programmen, für den sich der Mac ja dank seiner Grafikorientierung besonders anbietet, wurde wiederholt moniert. Bei integrierten Programmen wie »Excel« oder »Jazz«, stößt man zudem schnell an die Grenzen des vermeintlich großen Arbeitsspeichers. Der Einsatz des »Switcher«, mit dem sich individuelle Programme zu einer Art Integration zusammenfügen lassen, macht diese Tatsache nur um so deutlicher.

Anfänglich von einigen Anwendern fast als überdimensioniert empfunden, zeigen die Laufwerke des Macintosh mit ihren 400 KByte Speicherkapazität mitunter dann schneller als erwartet ihre Grenzen. Wer größere Datenmengen zu verwalten hat, kommt mit 400 KByte Speicherplatz pro Diskette nicht sehr lange aus.

Virtuose Bedienung mit mehr Tasten

Über die Tastatur und die Maus-Technik des Mac wurde viel geschrieben und gestritten. Ohne Cursortasten kann man nicht vernünftig arbeiten, hieß es da immer wieder. Und eine Tastatur ohne integrierten Zehnerblock sollte man für den Einsatz im Büro gar nicht anbieten.

So alt wie der Macintosh (ziemlich genau zwei Jahre) ist schließlich auch der Streit, ob ein Personal Computer »offen« sein muß, um ihn nachträglich mit Karten aufrüsten zu können, oder ob nicht eine geschlossene Box à la Mac, die regelmäßig durch Aufrüstung beziehungsweise Austausch von Hard- und/oder Software auf den jeweils höchsten technischen Stand gebracht wird, genauso gut ist.

Apple hat sich diese Wünsche und Diskussionen allem Anschein nach sehr genau angehört und darauf mit der Vorstellung des »Macintosh Plus« geantwortet. Primär für den kommerziellen Einsatz konzipiert, weist der MacPlus eine Reihe von Merkmalen auf, die ihn im Vergleich zu einer Vielzahl anderer PCs sehr gut abschneiden lassen.

Äußerlich sieht man dem MacPlus den Unterschied zur 512-KByte-Version fast nicht an: Lediglich der dezente Schriftzug und das bunte Firmenlogo auf der Frontseite verraten dem Eingeweihten, um welches Gerät es sich da handelt. Der eigentliche und

wichtigste Teil des Fortschritts verbirgt sich im Inneren des Gerätes: Der Arbeitsspeicher des MacPlus weist serienmäßig eine Kapazität von 1 MByte auf. Mit der Markteinführung der MBit-Chips, die demnächst ansteht, besteht durch ein einfaches Auswechseln der Speicherbausteine die Möglichkeit, die RAM-Kapazität bis auf 4 MByte auszubauen. Eine der Besonderheiten des MacPlus ist dabei, daß alle Programme auf diesen Riesenspeicher ohne Einschränkungen zugreifen können.

Anwender von integrierten Programmen wie Excel und Jazz brauchen sich damit keine Sorgen mehr zum Thema Größe des Arbeitsspeichers machen. Excel und Jazz vermögen bei 1 MByte RAM-Speicher ihre jeweiligen Stärken voll zur Geltung zu bringen. Was zunächst nach Programm-Gigantomanie aussah, kann jetzt seine Ernsthaftigkeit bei der kommerziellen Anwendung unter Beweis stellen. Wer demgegenüber die individuelle Integration mit Switcher bevorzugt, kann bis zu acht Programme gleichzeitig im 1-MByte-RAM-Speicher halten.

Die zweite wichtige Verbesserung beim MacPlus ist das Diskettenlaufwerk. Indem es 3,5-Zoll-Disketten nunmehr beidseitig beschreibt, beträgt nun die Speicherkapazität pro Diskette 800 KByte. Programme wie die bereits zitierten Excel, Jazz oder PageMaker finden damit einschließlich System- und Finderdateien auf nur einer Diskette Platz; bei Jazz zum Beispiel müssen für den Betrieb unerläßliche Systemdateien nicht mehr auf jede Datendiskette kopiert werden. Disketten- und Dateiverwaltung erfahren durch diese Neuerung eine wesentliche Verbesserung.

Reicher an Speicher

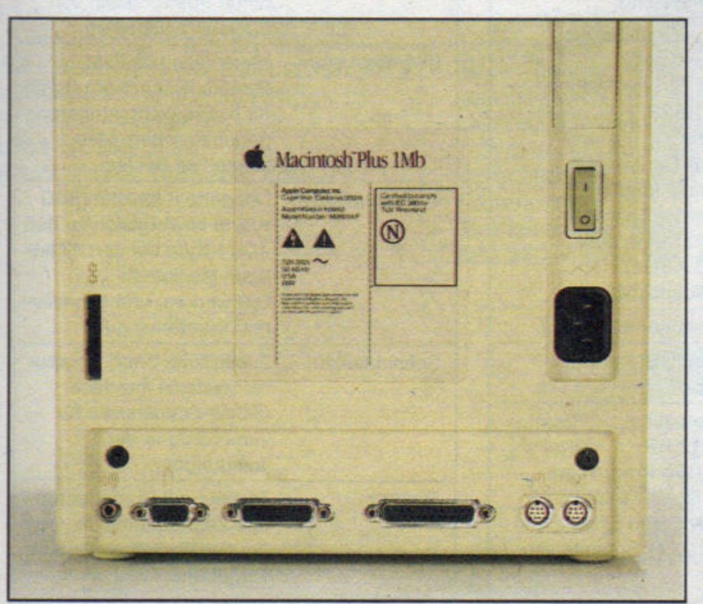
Neu ist auch ein externes 800-KByte-Disketten-Laufwerk, das nicht nur an den MacPlus paßt. Besitzer eines Macintosh mit 128 KByte und 512 KByte RAM-Speicher können so »neue« und »alte« Disketten parallel einsetzen. Umgekehrt besteht für Besitzer eines 400- KByte-Disketten-Laufwerks die Möglichkeit, es an den MacPlus anzuschließen. Dabei können die 800-KByte-Laufwerke die »alten« 400-KByte-Disketten lesen und (einseitig) beschreiben. Die Initialisierung neuer Disketten kann damit wahlweise einseitig (400 KByte) oder zweiseitig (800 KByte) erfolgen. Die 400-KByte-Laufwerke sind natürlich nur in der Lage, 400-KByte-Disketten zu lesen, zu beschreiben und zu formatieren.

Mit der verdoppelten Speicherkapazität bei den Disketten tritt naturgemäß ein Problem in den Vordergrund, das früher nur in extremen Fällen Anlaß zu Verärgerung gab. Rechnet man pro Schreibmaschinenseite mit etwa 1500 Byte Speicherbedarf, passen auf eine zweiseitig formatierte Diskette gut 500 solcher Seiten. Wer dann bei einer vollen Diskette nach einem bestimmten Dokument sucht, rauft sich schnell die Haare, denn im ungünstigsten Fall muß er 500 Namen durchlesen.

Aus dieser, aber auch aus verschiedenen anderen praktischen Erwägungen heraus, wird der MacPlus mit einer aktualisierten Systemdatei (Version 3.0) und einem weiter verbesserten Finder (Version 5.1) ausgeliefert. Darin ist ein Hierarchisches File System (HFS) enthalten, das nunmehr auch den Macintosh-Besitzern erlaubt, ihre Dateien mit Subdirectories abzulegen. Das heißt, man kann gleichartige Dateien in Gruppen zusammenfassen. Erfreulicherweise steht dieses HFS auch dann zur Verfügung, wenn man ausschließlich die einseitigen 400-KByte-Laufwerke im Einsatz hat.



Sowohl Cursortasten als auch Zehnerblock hat man auf der bisherigen Macintosh-Tastatur vergebens gesucht



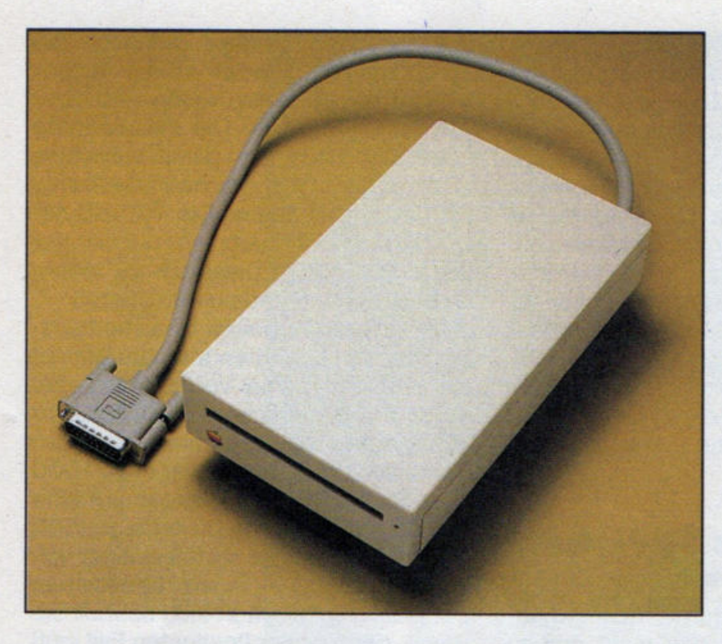
Zahlreiche Schnittstellen sorgen für schnellen und problemlosen Datenaustausch mit den Peripherie-Geräten

Famose Formatierung

Die 800-KByte-Laufwerke formatieren im übrigen nicht nur wahlweise ein- oder zweiseitig, sondern können zudem bei einseitiger Formatierung die zu initialisierende Diskette auf Tastendruck entweder auf das herkömmliche Macintosh File System (MFS) oder das neue HFS einrichten.

Im ersten Augenblick verwirrt diese Vielfalt vielleicht eher, als daß sie begeistert. Wenn man das Prinzip aber einmal durchschaut hat, gewährleistet es eine Flexibilität, wie man sie anderswo immer noch vergeblich sucht.

Das neue System und der verbesserte Finder arbeiten zudem deutlich schneller als die bisherigen Versionen. In Verbindung mit den doppelseitigen Laufwerken spricht Apple sogar von einer fünffach gesteigerten Arbeitsgeschwindigkeit.



Die neuen externen Macintosh-Laufwerke packen 800 KByte Daten auf eine Diskette

Der neue MacPlus besitzt auch eine erweiterte Tastatur, die endlich über einen Zehnerblock und Cursortasten verfügt. Wer mit dem abgesetzten Ziffernblock arbeitet, sollte jedoch darauf achten, daß das verwendete Programm die damit gemachten Eingaben auch »versteht«. Lotus bietet in diesem Zusammenhang eine aktualisierte Version ihres Programms Jazz 1A an, die auch sonst noch einige Vorteile zu bieten hat.

Betrachtet man den MacPlus einmal von der Rückseite, fallen zwei Dinge auf. Erstens gibt es eine SCSI-Anschlußbuchse, und zweitens wurden Drucker- und Modemverbindung neu gestaltet.

Mit dem SCSI-Anschluß am MacPlus geht Apple erstmals den Schritt in Richtung eines »offenen« Macintosh, auch wenn man die als »Small Computer Systems Interface« korrekt bezeichnete Buchse nicht fehlinterpretieren sollte. Sie ist für den Anschluß von Festplatten und Bandlaufwerken, Scannern und Digitizern gedacht, so daß Drucker- und Modembuchse für ihren eigentlichen Zweck freibleiben. Welche anderen Geräte an der SCSI-Buchse noch anzuschließen sein werden, bleibt abzuwarten. Die Fantasie der Fremdhersteller kennt da bekanntlich wenig Grenzen.

Alles in allem ein sehr attraktives Angebot, das Apple mit dem Macintosh Plus auf den Markt gebracht hat. Wer bereits einen Macintosh besitzt, braucht ihn übrigens nicht gleich zu verschrotten: Wie bei allen bisherigen Weiterentwicklungen lassen sich vorhandene Geräte auch dieses Mal preiswert aufrüsten.

Folgende Kombinationen stehen dabei zur Auswahl.

- Austausch des internen Laufwerks (400 KByte/einseitig) gegen das zweiseitige 800 KByte einschließlich neuer ROMs (128 KByte) für HFS
- Austausch der Hauptplatine (128 beziehungsweise 512 KByte RAM) gegen eine neue Hauptplatine mit 1 MByte RAM (später durch einfaches Chipwechseln bis auf 4 MByte erweiterbar)
- Austausch der vorhandenen Tastatur gegen die integrierte Tastatur mit Zehnerblock und Cursortasten

Wer lieber bei seinem vertrauten Macintosh bleibt, kann den Ausbau auch alleine auf das neue Betriebssystem und den neuen Finder beschränken. Dazu stellt Ihnen Ihr Fachhändler nach Aussage von Apple eine Kopie vom Aktualisierungsprogramm zur Verfügung. Der Macintosh Plus ist eine gelungene Weiterentwicklung des bewährten Macintosh-Konzepts.

(Peter Kraft/ts)

Technische Daten Macintosh Plus

Prozessor	MC 68000
Arbeitsspeicher	1 MByte, ohne Tausch der Hauptplatine unter Verwendung von 1 MBit-Speicher-Chips bis auf 4 MByte erweiterbar
Massenspeicher	3,5-Zoll-Diskettenlaufwerk mit wahlweise doppelseitiger (800 KByte) oder einseitiger Formatierung (400 KByte),
ROM	128 KByte, enthält Teile des Betriebssystems, AppleTalk und den neuen hierarchischen Finder (HFS)
Bildschirm	hochauflösender Monochrom-Bildschirm, 512x342 Punkte
Tastatur	Volltastatur mit Schreibfeld (deutscher Zeichensatz), Zehner-Block, Cursor-, Wahl- und Befehlstasten
Betriebssystem	Fenster- und Maustechnik, Rollmenü, Schreibtischoberfläche mit Datei- und Programmsymbolen, hierarchische Dateiverwaltung (Ordnersystem)
Schnittstellen	1 x Maus/Grafiktablett-Schnittstelle 2 x RS422 (vergleichbar mit RS232) 1 x SCSI Small Computer System Interface 1 x externes Laufwerk (400 KByte oder 800 KByte) 1 x Audioanschluß
Ton	Tongenerator eingebaut
Uhr/Kalender	eingebaut, Batterie-gepuffert
Software	voll kompatibel zu Macintosh 512 KByte weltweit über 1000 Programme, davon ist ein Großteil für kommerziellen Einsatz und in deutscher Sprache verfügbar

Vorteile zur bisherigen Macintosh-Version

Arbeitsspeicher	Standardmäßig 1 MByte statt 128 oder 512 KByte, auf 4 MByte aufrüstbar durch Auswechseln von Speicherbausteinen
Betriebssystem	Hierarchisches File-System mit Ordnern (HFS) für bessere Strukturierung gegenüber dem Macintosh File System
Diskettenstation	Doppelte Kapazität (800 KByte im Vergleich zu den 400 KByte bei den einseitigen Macintosh-Laufwerken) und schnellerer Diskettenzugriff
Schnittstellen	Zusätzliche Small Computer Systems Interface (SCSI)-Schnittstelle für verschiedene Verwendungen
Tastatur	Tastatur mit Cursortasten und Zehnerblock, die man bei der Macintosh-Tastatur vergebens sucht

Ein offenes Tor zur Außenwelt

Steuerungen sind ein interessantes Gebiet der Computeranwendung. Leider wurde dem Atari ST das nötige Rüstzeug nicht mit in die Wiege gelegt. Eine Parallelschnittstelle jedoch macht's möglich.

Der Atari ST verhält sich von den Schnittstellen her recht »kontaktfreudig«. So finden sich alle nur erdenklichen Anschlüsse für Maus, Joystick, RS232, Drucker und Harddisk. Aber alle diese Schnittstellen erfüllen eine Anforderung nicht: Will man damit auf einfache Weise externe Geräte steuern (zum Beispiel die Modelleisenbahn), so erfordern sie einen großen Hardwareaufwand, um die entsprechenden Signale zu erzeugen. Eine Ausnahme bildet der Drukkerport. Er stellt acht Datenleitungen sowie eine Steuerleitung zur Verfügung. Die wenigsten Atari-Besitzer wissen über die bidirektionale Nutzbarkeit dieser Schnittstelle Bescheid. Im Klartext heißt das, es lassen sich auch Daten von »außen« empfangen.

Die Programmierung dieser Schnittstelle ist von Basic aus recht einfach. Hohe Schaltgeschwindigkeiten jedoch erzielen Sie damit nicht. Außerdem stehen immer nur acht Bit zur Verfügung, so daß man bei komplexeren Aufgaben wieder zum Lötkolben greifen muß.

Eine Programmierung auf Assemblerebene bringt zwar einen enormen Zuwachs an Geschwindigkeit, löst aber das Problem der geringen Datenbreite nicht.

Ebenso verhält es sich mit den übrigen Schnittstellen. So bietet die Midi-Buchse serielle Übertragung mit mehr als 31 Kilobaud, aber die Dekodierung der Signale ist ebenfalls sehr aufwendig. Über die DMA-Buchse lassen sich bei geeigneter Programmierung auch Daten verschicken, doch auch hier gestaltet sich der Aufwand an zusätzlichen Bauteilen recht hoch.

Kleiner Aufwand große Wirkung

Der letzte Weg nach außen führt über den ROM-Modulport. Dort wurden Adressen- und Datenleitungen sowie verschiedene Steuerleitungen herausgeführt. Allerdings ist der Zugriff auf die Außenwelt nicht einfach. Durch seine eigentliche Funktion (Programme in ROMs) kann man von diesem Port nur lesen. Die fehlende Schreib-/Leseleitung läßt sich auch wieder nur mit entsprechendem Bastelaufwand erzielen.

Abhilfe schafft hier der 32-Bit-Parallelport, eine kleine Platine, die einfach in den ROM-Modulport des Atari gesteckt wird.

Diese Erweiterungskarte bietet mehr als nur eine bidirektionale, 32 Bit breite, Parallelschnittstelle. Durch sogenannte »Jumper« (kleine Kurzschlußstecker) läßt sich die Karte an nahezu jede Aufgabe anpassen. So ändert schon ein einfaches Umstecken den Adreßbereich, in dem der Port liegen soll. Auch die Funktionen der einzelnen 8-Bit-Ports lassen sich mit Jumpern leicht umgestalten.

Will man die Schnittstelle in ihrer ursprünglichen Form (externe ROMs) betreiben, kann die Steckkarte an ihrem Platz bleiben. Auch für diese Umschaltung sorgt ein Jumper. Was fehlt, ist lediglich ein oder mehrere Sockel für die ROM-Bausteine.

Bei der Entwicklung dieser Schnittstelle wurde dies berücksichtigt. Am Ende der Karte findet man ein Lochrasterfeld, das mehrere Sockel oder andere Erweiterungen aufnimmt.

Die Programmierung dieser Schnittstelle ist sowohl von Basic als auch von Assembler aus möglich. Mit zum Lieferumfang gehört ein gut dokumentiertes Basic-Listing mit den einzelnen Schritten zur Programmierung dieser Schnittstelle.

So fällt es nicht schwer, diesen Algorithmus in andere Sprachen (C, Modula oder Assembler) zu übersetzen. In den erwähnten Sprachen kommt man in den schon erwähnten Genuß einer höheren Geschwindigkeit.

Diese Schnittstelle erfüllt auf jeden Fall viele Bastlerträume. Einem EPROM-Programmiergerät steht jetzt ebensowenig im Wege wie der Steuerung der Modelleisenbahn. Oder schützen Sie doch Heim und Hof mit Ihrem Atari als Alarmanlage. Mit der Schnittstelle lassen sich bis zu 32 Türen, Fenster und sonstige Gegenstände überwachen. Die Hardware zur Verwirklichung all dieser Ideen ist mit dem 32-Bit-Parallelport gegeben. Dies sind nur einige Beispiele von vielen.

(Udo Reetz/lg)

Schon wenige Bauteile machen den Atari ST noch kontaktfreudiger

BNT Computerfachhandel GmbH, Marktstraße 4, 7000 Stuttgart 50

»Sidecar«: MS-DOS auf dem Amiga

Für IBM-kompatible Computer existiert eine unüberschaubare Menge an leistungsfähigen Programmen für die verschiedensten Anwendungsbereiche. Diese Software-Quelle erschließt ein MS-DOS-Emulator nun auch dem Amiga.

Der Amiga ist zwar der Star in Sachen fortschrittlicher Computertechnologie, aber er leidet, wie alle völlig neuen Computer, an Software-Mangel. Was lag da näher, als das Problem mit einem MS-DOS-Emulator zu lösen: Erstens deckte man so den Markt der MS-DOS-Kompatiblen ab und sorgte zweitens für Software in rauhen Mengen. Entwickelt wurde »Sidecar«, so der vorläufige Name der Erweiterung, unter der Leitung deutscher Commodore-Spezialisten, die dabei auf ihr PC10-Wissen zurückgreifen konnten.

Flotter Beiwagen

Beim PC 10 handelt es sich um einen reinen MS-DOS-Personal-Computer von Commodore. Ab Herbst dieses Jahres soll der Emulator auch in unseren Landen erhältlich sein.

Sidecar, das unscheinbare Kästchen, läßt nicht erahnen, daß darin ein voll funktionstüchtiger MS-DOS-Computer steckt. Auf zwei Platinen sind ein mit 4,77 MHz getakteter 8088-Prozessor, 256 KByte RAM, ein Floppy-Controller sowie ein Sockel für den Arithmetik-Prozessor 8087 untergebracht, der den Hauptprozessor bei zeitintensiven Rechenaufgaben entlastet. Der Speicher ist auf der Platine mit 256-KBit-Chips auf 512 KByte erweiterbar.

Drei Steckplätze für IBM-kompatible Steckkarten der vollen Baulänge bieten optimale Voraussetzungen für eine Aufrüstung des Emulators. Vorstellbar und empfehlenswert ist beispielsweise eine Festplatten-Karte, da auch der Amiga Zugriff auf die »Sidecar«-Hardware hat und somit die Harddisk mitbenutzen



Der »Sidecar«-Anschluß ist problemlos: Der Emulator wird einfach an den Amiga angesteckt

kann. Über eine Steckerleiste besteht ferner die Möglichkeit, den Speicher des Amiga auf 2 MByte aufzustocken. Diesem Anschluß kommt gerade deshalb besondere Bedeutung zu, da der Amiga-Expansionsport nicht durchgeschleift ist und Sidecar somit keine anderen Erweiterungen mehr zuläßt.

Bis zu zwei Laufwerke mit einer formatierten Speicherkapazität von 360 KByte verwaltet der Floppy-Controller. Ein $5^1/_4$-Zoll-Laufwerk ist bereits in »Sidecar« eingebaut, ein Zweitlaufwerk findet Anschluß an der Gehäuse-Rückseite. Für die Stromversorgung sorgt ein eingebautes Netzteil, für dessen Kühlung ein Lüfter zuständig ist. Computer und Emulator benötigen übrigens nur ein Netzkabel, da eine Stromleitung von »Sidecar« zum Amiga führt.

Angeschlossen wird »Sidecar« rechts an den Amiga-Expansionsport. Die dadurch verdeckten Maus- und Joystickanschlüsse stehen ebenfalls mit der Erweiterung in Verbindung und befinden sich jetzt an der »Sidecar«-Vorderseite.

Die eigentliche Neuentwicklung bei »Sidecar« ist das Interface, das den Datenaustausch zwischen dem 68000-System Amiga und dem Emulator regelt, der auf dem 8088-Prozessor basiert. Die Commodore-Entwickler standen vor dem Problem, Daten vom Amiga zu »Sidecar« und die Bildschirmausgaben (Texte und Grafik) zum Amiga zu transportieren. Die Übertragung erfolgt über ein 128 KByte-RAM, auf das sowohl der Amiga als auch »Sidecar« zugreifen.

Um das Amiga/»Sidecar«-Gespann in Betrieb zu nehmen, laden Sie auf dem Emulator MS-DOS 2.11, auf der Seite des Amiga entweder »PC Mono« oder »PC Color«. Diese zwei Programme können sowohl einen Monochrom-Bildschirm als auch eine Farbausgabe simulieren. Die Programme übernehmen den Datenaustausch zwischen dem Amiga und dem »Sidecar«-Computer. Alle Eingaben auf der Amiga-Tastatur überträgt das Programm zu »Sidecar«, alle Bildschirmausgaben des Emulators erscheinen auf dem Amiga-Bildschirm, je nach Programm in Farbe oder monochrom.

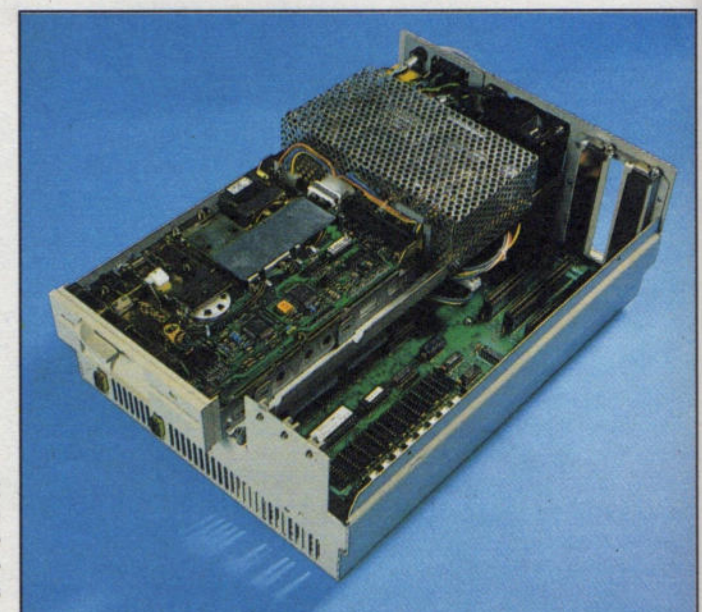
Nach dem Laden eines dieser beiden Programme zeigt der Amiga-Monitor einen neuen Screen mit einem Fenster, speziell für alle »Sidecar«-Ausgaben. Der Screen läßt sich beliebig nach unten wegziehen, und das Fenster ist frei in der Größe veränderbar. Per Menü kann man auch den Fensterrand verschwinden lassen, um den Eindruck eines »echten« Personal-Computer-Monitors noch zu verstärken.

Das Multitasking-Betriebssystem versetzt den Amiga in die Lage, neben der MS-DOS-Software auch noch eigene Programme auszuführen. Ein Verschieben des MS-DOS-Screens gibt den Blick auf die Ausgaben der Amiga-Software frei. Die Ablaufgeschwindigkeit der MS-DOS-Programme behindert zusätzliche Amiga-Software in keiner Weise, da »Sidecar« als eigenständiger Computer nur die Tastatur-Eingaben vom Amiga holt und seine Bildschirmdaten auf dem Amiga ausgibt.

Dadurch, daß sich »Sidecar« nicht um diese Ein- und Ausgaben zu kümmern braucht, arbeitet die gesamte Software sogar noch etwas schneller als auf dem Original-IBM-Personal-Computer. Erstklassig präsentiert sich »Sidecar« auch in Sachen Kompatibilität: Weder von Wordstar noch vom Flugsimulator ist der Emulator aus dem Gleichgewicht zu bringen.

Das »Sidecar«-Konzept besticht sowohl von der Soft- als auch von der Hardwareseite durch einen ausgereiften und leistungsfähigen Aufbau. Dieser Emulator besitzt die besten Voraussetzungen zu einem großen Erfolg – vorausgesetzt, der Preis stimmt.

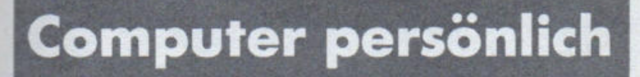
(ts)



Unter dem Laufwerk und Netzteil sehen Sie die Hauptplatine mit den Steckplätzen für RAMs und Steckkarten



Auf Ihrem Weg zum professionellen Computer-Anwender werden Sie früher oder später ...



Computer persönlich

Das aktuelle Fachmagazin für Personal Computer.

■ Wenn Sie jetzt den Schritt vom Heim-Computer zur professionellen Anwendung eines Personal Computers planen ■ Wenn Sie beruflich oder privat bereits einen Personal Computer benutzen ■ Wenn Sie selbst professionell programmieren ■ Wenn Sie regelmäßig Informationen über das breite Produktangebot auf dem Personal Computer-Markt benötigen ■ Wenn Sie professionelle Hard- und Softwaretests suchen ■ Wenn Sie Ihr eigenes System möglichst effizient einsetzen wollen, dann ist »Computer persönlich« genau Ihre Zeitschrift.

Die konsequente Ausrichtung auf professionelle Anwendungen bietet Ihnen alle wichtigen Informationen.

Und das alle 14 Tage, mittwochs bei Ihrem Zeitschriftenhändler oder im Computer-Fachgeschäft. ■

PC Magazin

Die einzige Wochenzeitung ausschließlich für Personal Computer im IBM-Standard.

■ Wenn Sie an aktuellen und umfassenden Informationen über IBM-PCs und kompatible Systeme interessiert sind ■ Wenn Sie stets über die neuesten und effektivsten Anwendungen für den professionellen und privaten Bereich informiert sein wollen ■ Wenn Sie sich für Marktübersichten und ausführliche Testberichte über Hard- und Software interessieren ■ Wenn Sie sich mit CAD/CAM, Netzwerken und der Anbindung von PCs an Groß-EDV-Anlagen beschäftigen, dann ist »PC Magazin« genau auf Ihre Bedürfnisse zugeschnitten.

Die Spezialisierung auf IBM-PCs und Kompatible ermöglicht eine gezielte Berichterstattung und bietet genügend Raum, um auf Anwenderprobleme spezifisch eingehen zu können.

»PC Magazin« — jeden Mittwoch neu bei Ihrem Zeitschriftenhändler oder im Computer-Fachgeschäft. ■

... auf diese beiden Computer-Zeitschriften stoßen:

Zur Anforderung Ihres kostenlosen Probeexemplares einfach den nebenstehenden Gutschein ausfüllen, ausschneiden, auf eine Postkarte kleben oder in ein Kuvert stecken und einsenden an: **Markt & Technik, Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar bei München.**

GUTSCHEIN

für ein kostenloses Probeexemplar

Senden Sie mir die neueste Ausgabe der von mir angekreuzten Zeitschrift kostenlos als Probeexemplar.

☐ Wenn mir »Computer persönlich« zusagt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte »Computer persönlich« dann regelmäßig alle 14 Tage per Post frei Haus geliefert und bezahle pro Jahr nur DM 98,– statt DM 143,– Einzelverkaufspreis. Zustellung und Postgebühren übernimmt der Verlag. Dieses Angebot gilt nur in der Bundesrepublik Deutschland einschließlich West-Berlin. Das Abonnement verlängert sich nur dann zu den dann jeweils gültigen Bedingungen, wenn es nicht 2 Monate vor Ablauf schriftlich gekündigt wird.

☐ Wenn mir das »PC Magazin« zusagt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte mein »PC Magazin« dann regelmäßig jede Woche per Post frei Haus geliefert und bezahle pro Jahr nur DM 155,– statt DM 229,50 Einzelverkaufspreis. Zustellung und Postgebühren übernimmt der Verlag. Dieses Angebot gilt nur in der Bundesrepublik Deutschland einschließlich West-Berlin. Das Abonnement verlängert sich nur dann zu den dann jeweils gültigen Bedingungen, wenn es nicht 2 Monate vor Ablauf schriftlich gekündigt wird.

Name, Vorname

Straße

PLZ Ort

Datum 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs. Ich bestätige dies durch meine 2. Unterschrift.

Datum 2. Unterschrift

HCS986



Die Traumfabrik

Das Abbild der Umwelt wirklichkeitsgetreu in den Computer zu übertragen und dann bearbeiten zu können, ist das Ziel der Digitalisierung. Wie nahe sind wir diesem hochgesteckten Ziel?

Digitalisieren ist für Computerfreaks ein Zauberwort, das ihre Herzen höher schlagen läßt. Ein reales Bild in den Computer zu übertragen und dort nach Lust und Laune zu manipulieren, fasziniert viele. Obwohl zwischen dem Realbild und dem Computerbild ein Unterschied klafft, ist man heute bereits mit relativ preiswerten Mitteln in der Lage, Beachtliches zu leisten.

Was bedeutet eigentlich »Digitalisieren«? Global ausgedrückt ist darunter die Umsetzung eines analogen Signals in digitale Werte zu verstehen. Eine handelsübliche Videokamera liefert einem Analog-Digital-Wandler ein analoges Signal, aus dem dieser digitale Werte produziert. Erst diese digitalen Werte sind für den Computer verwertbar (Bild 1). Dabei entscheiden zwei Kriterien über die Wirklichkeitsnähe des daraus entstehenden Bildes. Zum einen die Auflösung des Meßwerts. Sie ergibt sich aus der Menge der Bits, die der Computer pro Messung für das Abspeichern zur Verfügung stellt. Je mehr Speicherplatz der Computers aufweist, desto mehr Bits kann er pro umgesetzten Bildpunkt auswerten, und um so exakter kann er den tatsächlichen analogen Meßwert darstellen. Die Software von guten Digitizern arbeitet mit 8 bis 10 Bit. Bei 10 Bit läßt sich der gesamte Meßbereich in 1024 Schritte unterteilen.

Das zweite Kriterium ist die Meßgenauigkeit des A/D-Wandlers. Er muß das Signal so genau wie möglich be-

stimmen, um dann bei der Wiedergabe die richtigen Werte zu reproduzieren.

Ein analoges Signal wird also vom A/D-Wandler in digitale Werte »zerteilt«. Aus einer Sinus-Kurve ensteht so eine Zahlenkolonne, bei der Synthese in einem D/A-Wandler schließlich eine Treppe. Der Meßbereich heutiger Computer ist immer noch viel zu klein, die vielen Nachkommastellen eines analogen Wertes zu berücksichtigen. Er muß den Meßwert runden. Je kleiner aber die Rundung, desto präziser gleicht die reproduzierende Treppe der ursprünglichen Kurve und desto originalgetreuer wirken die Farbverläufe.

Schneller als ein Augenblick

Jedes Videobild wird sehr schnell ausgebaut. Das muß so schnell gehen, um das menschliche Auge nicht merken zu lassen, daß es aus lauter sehr kurzlebigen Einzelbildern besteht. Alle $^1/_{25}$ Sekunden folgt ein neues Bild. Um ein solches Bild, das nur einen Augenblick auf dem Bildschirm erscheint, abzutasten, wäre ein großer Hard- und Softwareaufwand nötig. Aber es muß ja nicht genau dieses eine Bild sein. Das Bild darf wechseln, wenn der Inhalt für einige Sekunden der gleiche ist. Digitizer behelfen sich mit diesem einfachen, aber effektiven Trick. Voraussetzung dafür ist ein statisches Motiv. Richtet man die Videokamera auf eine Person, muß sich diese eben für zirka fünf Sekunden ruhig verhalten.

Bei manchen Geräten wird das Bild scheinbar direkt auf den Monitor des Computers übertragen. In Wirklichkeit aber liefert die Kamera das analoge Bild, produziert der A/D-Wandler daraus mit Hilfe von Software die digitalen Werte, leitet diese Werte umgehend an einen D/A-Wandler in seinem Inneren und produziert wieder ein analoges Bild. Denn nur auf diesem Weg kann der Computer ein Bild auf seinem Monitor wiedergeben. A/D-Wandler, Computer und D/A-Wandler zwischen der Kamera und dem Monitor wirken wie »Digitalfilter«. Beim Enstehen des Bildes fällt auf, daß es nicht zeilenweise, sondern spaltenweise auf dem Monitor aufgebaut wird. Das liegt an der Abtastung.

Der A/D-Wandler richtet sich nach dem Zeilensprungsignal, das er für die Abtastung einsetzt. Der eigentliche Abtastimpuls entsteht durch das um einen bestimmten Betrag verzögerte Zeilensprungsignal. Verzögert man immer um den gleichen Betrag, dann mißt der A/D-Wandler die Helligkeit desselben Punktes einer Zeile und gibt den Wert an den Computer weiter, der ihn in seinem Speicher ablegt. So macht er das Zeile für Zeile. Bei der deutschen Fernsehnorm sind das genau 625 Zeilen. Die Meßpunkte liegen alle genau untereinander. Das ergibt dann die Werte einer Spalte. Die Verzögerung zum Zeilensprung ist variabel und wird vom Programm gesteuert. Wurde eine komplette Spalte gemessen, erhöht das Programm die Verzögerung um einen kleinen Betrag für den Zeitraum eines vollen Bildes. Dadurch erhält es die Helligkeitswerte der folgenden Spalte und speichert sie im Computer. Man benötigt keinerlei Synchronisierung, da das Taktsignal vom Videobild selbst stammt. Der Hardwareaufwand bleibt durch diesen Umstand relativ gering. Aus diesem Grund und der immer billiger werdenden Videotechnik und Elektronik sinkt der Preis für Digitizer immer mehr. Die resultierenden Bilder wirken trotz der Einschränkungen, die man noch machen muß, bereits beeindruckend realistisch, wie unsere Fotos belegen.

Die Mischung macht's

Ein weiteres Kriterium für die Qualität eines Bildes ist die Farbtreue. Jede Farbe kann aus einer Mischung von Rot, Grün und Blau zusammengesetzt werden. Durch die Mischung dieser drei Spektralfarben entstehen alle anderen Farben. Je feiner der Compu-



Bild 2. Leider nur digitale Vision: Auto und Umwelt in trauter Harmonie

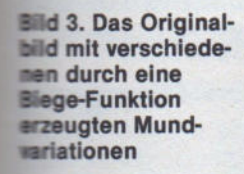


Bild 3. Das Originalbild mit verschiedenen durch eine Biege-Funktion erzeugten Mundvariationen





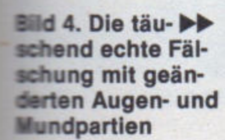


Bild 4. Die täuschend echte Fälschung mit geänderten Augen- und Mundpartien ▶▶

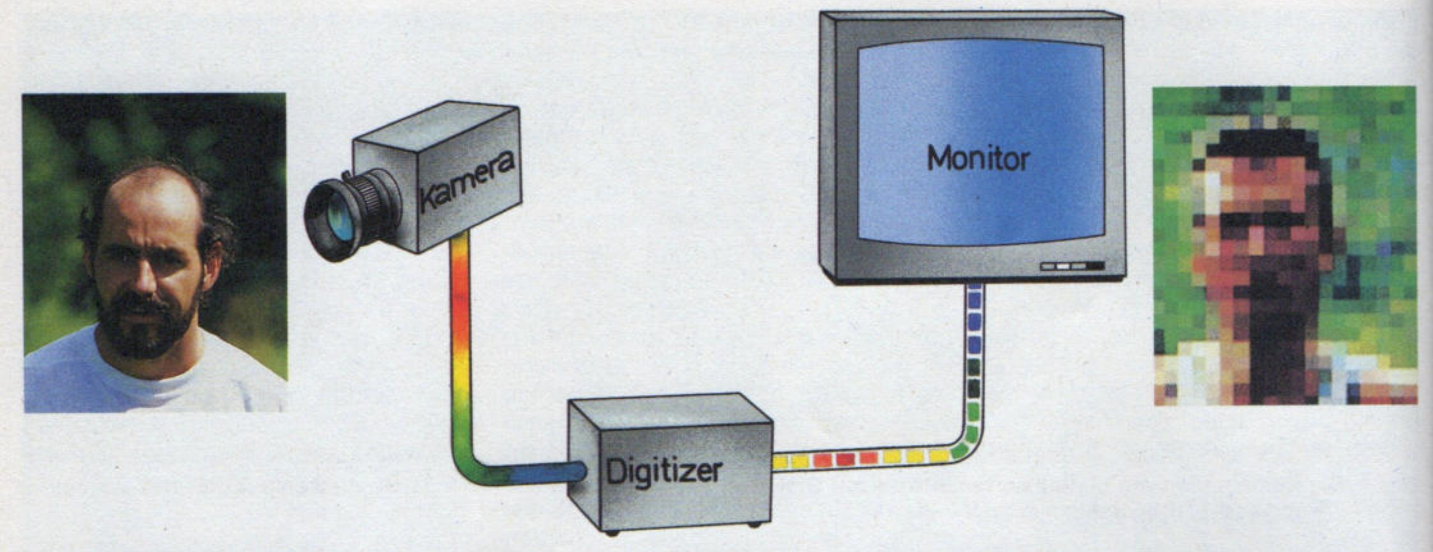


Bild 1. Der Weg eines digitalisierten Bildes: Von der Kamera zum Digitizer in den Monitor

ter diese Grundfarben abstufen kann, desto realer erscheint das von ihm produzierte Bild. Selbst so fantastische Grafikkünstler wie der Amiga bringen es aber »nur« auf je 16 Abstufungen. Das menschliche Auge vermag aber viel mehr Stufen zu unterscheiden. Aus seinen 16 Farbwerten je Grundfarbe produziert der Amiga immerhin 4096 verschiedene Farben. Diese Zahl ergibt sich aus 16 verschiedenen Rottönen multipliziert mit 16 verschiedenen Blautönen und 16 verschiedenen Grüntönen.

Mehr als nur Natur

Die möglichst naturgetreue Wiedergabe von Bildern ist aber nicht der eigentliche Zweck einer Bilddigitalisierung mit dem Computer. Das kann jede billige Fotokamera zehnmal besser. Was man mit den Bildern machen kann, wenn sie erst im Computer gespeichert sind, darin liegt der eigentliche Reiz. Betrachten Sie sich die Bilder. Das Auto im Wald erinnert an ein Werbeplakat für Katalysatoren. Nehmen wir an, es wurde von einer Werbeagentur entworfen. Zuerst wird der Wald digitalisiert, anschließend das Auto (Bild 2). Eine ganze Bibliothek mit Grundmotiven kann man sich auf diese Weise zusammenstellen.

Zahllose Autotypen und Hintergrundbilder jeder Art lassen sich sammeln und so das geeignete Bilderpaar kombinieren. Oder ein See gefällig? Holen Sie sich einen See von der Diskette. Das kann man auch mit Fotos? Richtig, aber was, wenn der Kunde ein kleineres Auto wünscht oder eine andere Lackfarbe? Für den Computer mit dem richtigen Programm kein Problem. Bitte, ein Tastendruck genügt, und aus schneeweiß wird jede andere gewünschte Farbe. Das sind Banalitäten für ein gutes Programm. Wirklich Verblüffendes leistet der Computer bei Formmanipulationen. Das Mädchenbild zeigt es deutlich. Der Mund wurde als Ausschnitt herauskopiert und nach oben und unten in verschiedenen Stärken gebogen. Er ließe sich auch verkleinern oder vergrößern oder die Lippenfarbe ändern. Der Kosmetiksalon im Computer (Bild 3/4). Sogar ganze Bilder zu verkleinern oder vergrößern ist eine Sache weniger Sekunden (Bild 5). Auf einfache Weise kombiniert man Digitales mit Gezeichnetem. Möchten Sie eine Postkarte mit Ihrem Konterfei? Kein Problem. Aus verschiedenen Linienstärken, Schriftarten und Farben setzt man das gewünschte innerhalb von Minuten zusammen. Man muß sich nicht mit drei Vorschlägen begnügen, denn das Bild zu verkleinern, vergrößern, zu drehen und zu spiegeln oder mit anderen zu kombinieren, kostet nicht mehr als



Bild 8. Durch Farben (links) oder Äquidensiten (rechts) heben sich Strukturen ab



Bild 9. Farbige Solarisation oder Negativ – in der herkömmlichen Fototechnik mühsam – ist ein digitales Kinderspiel



Bild 6. Die Kombination von digitalisierten Bildern mit Grafik kann aufwendige Layouttechniken sparen helfen



Bild 7. Viele Graustufen bedeuten wenig Auflösung, wenige Graustufen aber eine hohe Auflösung



Bild 5. Vergrößern, verkleinern: Problemlos

einen Tastendruck oder einen Mausklick. Ein Beispiel zeigt Bild 5. Ausgefallenes Briefpapier, Werbeschreiben oder das Layout einer ganzen Zeitschrift inklusive Bilder zu entwerfen, ist bald mit dem Computer und einem geeigneten Programm ein Kinderspiel. Dabei ist ganz entscheidend, daß für die Kreativität mehr Zeit bleibt. Man kann sich auf das Wesentliche konzentrieren.

Fantasie ohne Grenzen

Der Fantasie sind keine Grenzen gesetzt. Um zu zeigen, wie naturgetreu man ein solches Bild retuschieren kann, betrachten Sie bitte Bild 4. Würden Sie erkennen, was alles gegenüber dem Original (Bild 3) einer Retusche unterzogen wurde? Seien Sie ehrlich, hätten Sie überhaupt gemerkt, daß etwas retuschiert wurde? Dabei helfen verschiedene Zoomfaktoren guter Programme. Durch die vielfache Vergrößerung sind äußerst detaillierte Veränderungen machbar.

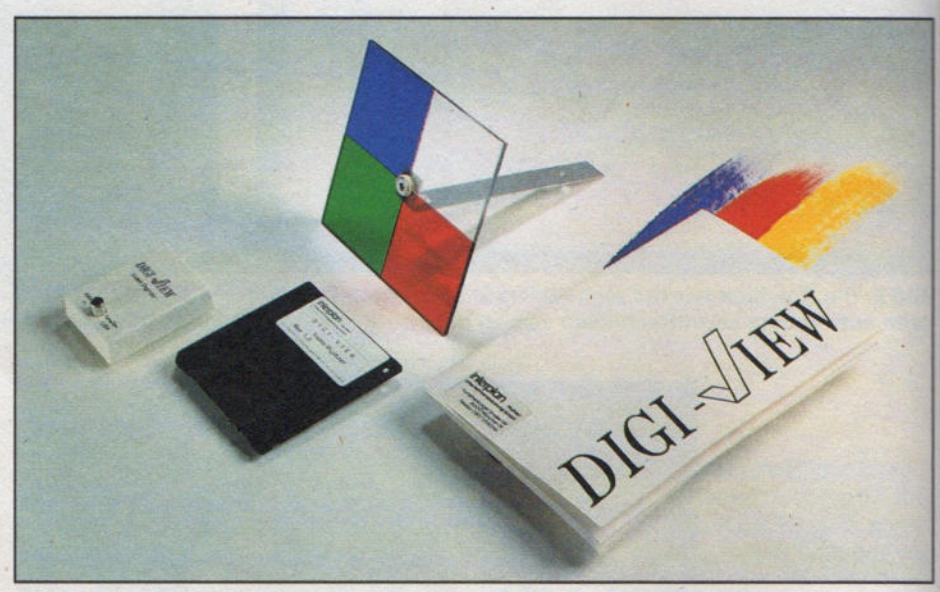
Aber was diese Bilder so plastisch macht, ist nicht nur die Auflösung – es sind die vielen Farben. Je mehr Farben für ein Bild zur Verfügung stehen, desto echter wirkt es. Das macht Bild 6 deutlich. Die Auflösung ist mit 320 mal 200 Punkten eigentlich niedrig, aber 4096 Farben zeigen ganz deutlich ihre Wirkung. Gerade Farben spielen bei vielen Anwendungen eine ganz große Rolle, denn damit läßt sich vieles zeigen, was man sonst nicht so leicht erkennen würde. Schattierungen, sonst kaum zu erkennen, treten durch kontrastreiche Farben überdeutlich hervor. Diese Technik der synthetischen Kontrastverstärkung wird heute in der Medizin eingesetzt. Röntgenbilder, deren Grauwerte manchmal so eng beieinanderliegen, daß sie vom menschlichen Auge nicht mehr zu unterscheiden sind, werden mit Falschfarben kontrastreicher gemacht. Genau dasselbe passiert mit Satellitenfotos. Die Atmosphäre wirkt im unbearbeiteten Bild wie dicker Nebel. Durch die richtige Farbwahl kann man diesen Nebel entweder durch Farbverstärkung beseitigen, dann entstehen naturgetreue, besonders klare Bilder, oder durch Falschfarben bestimmte Grauwerte ersetzen und auf diese Weise geographische Strukturen hervorheben. Auf diese Weise lassen sich zum Beispiel Mineralienvorkommen lokalisieren. Bei geeigneter Farbwahl treten Gebiete mit Mineraliengehalt kontrastreich hervor und können präzis abgegrenzt werden. Mit Hilfe von Computern und – neben anderen Methoden – dieser Technik wußten Wissenschaftler, lange bevor die Astronauten von Apollo 11 den Mond betraten, nicht nur, daß es über hohe Mineralvorkommen verfügt, sondern konnten sie genau bestimmen. Aufgrund solcher »Ferndiagnosen« wurde die Zusammensetzung fast aller Planeten unseres Sonnensystems analysiert. Eine Sonderform hebt Zwischenwerte der Grauskala hervor, die bei geeigneter Wahl eine »Wetterkarte« des Motivs bilden. Solche »Äquidensiten«-Effekte zeigen die Bilder 8 und 9. Sie gleichen farbigen Solarisationen der herkömmlichen Fototechnik.

Computer als Kreativitätsverstärker

Digitalisieren ist also keine Spielerei, sondern eine Anwendung, die in Zukunft immer weiter verfeinert wird. Bereits heute ist sie ein Hilfsmittel, auf das man in so unterschiedlichen Bereichen wie Wissenschaft und kreativen Berufen nicht mehr verzichten kann. Aber die Preise rutschen immer tiefer. Auch für den Privatanwender wird dieses neue Medium dadurch interessant. Preiswert auch, weil viele ohnehin eine Videokamera und einen Computer besitzen und nur noch den A/D-Wandler und die Software benötigen. (hb)

Amigas digit

Ein unscheinbares Kästchen, kaum größer als eine Streichholzschachtel, zaubert auf Ihren Amiga Bilder in bestechender Qualität mit brillanten Farben und hoher Auflösung.



Das »Digi-View«-Set: Digitalisierer, Software, Farbfilter und Anleitung

Daß sich der Amiga durch seine hervorragende Grafikdarstellung auszeichnet, ist bereits bekannt. Doch trotz seiner Fähigkeiten ist es schwer, mit ihm realistische Bilder zu zeichnen. Hier muß Elektronik der Elektronik helfen! Was liegt da näher, als Bilder von einer Videokamera in ein für den Computer verständliches Format zu übersetzen. Diese Aufgabe übernimmt ein kleines Modul, das an den Druckerport des Amiga gesteckt wird. Der Digitizer »Digi-View« ist der erste seiner Art für den Amiga. Er wird komplett mit Software und Farbfiltern ausgeliefert.

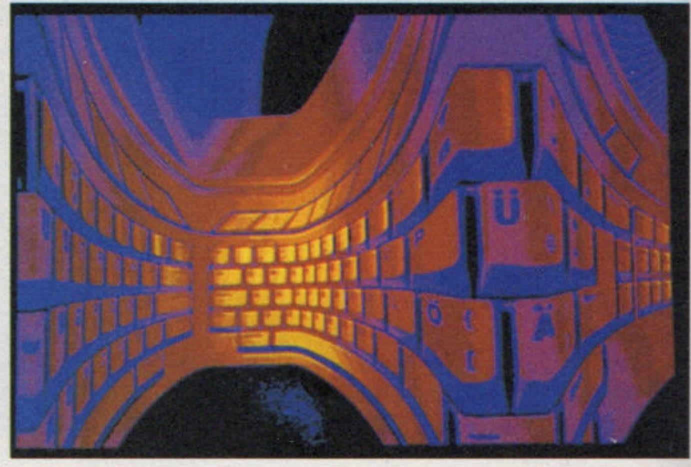
Um Bilder zu digitalisieren, benötigt man eine Videokamera, wie sie für alle gängigen Videosysteme erhältlich ist. Für unsere Zwecke eignen sich am besten Schwarzweiß-Überwachungskameras, die sich zudem preislich in akzeptablen Grenzen halten. Über ein Koaxialkabel mit Chinch-Stecker findet die Kamera Anschluß an den Amiga. Wenn Sie eine Farbkamera mit einem FBAS-Ausgang verwenden, benötigen Sie einen elektronischen Farbfilter, der die störenden Farbsignale herausfiltert. Allerdings führen einen solchen 4,43-MHz-Tiefpassfilter nur die wenigsten Fachgeschäfte. Da die meisten Farbkameras zudem mit einer geringeren Auflösung arbeiten als die preiswerten Schwarzweiß-Kameras, sind sie nur bedingt zu empfehlen.

Das Besondere am Digitalisierer ist, daß man auch ohne Farbkamera ein farbiges Bild erhält. Bekanntlich nutzt jedes Fernsehgerät nur die drei Grundfarben Rot, Grün und Blau, und erzeugt durch Farbmischung und Überlagerung die restlichen Farben. Wenn man nun von einem farbigen Gegenstand in drei aufeinanderfolgenden einfarbigen Digitalisierungen durch Farbfilter hindurch jeweils nur den Rot-, Grün- und Blauanteil aufnimmt, kann der Computer aus den drei Bildern ein farbiges Gesamtbild errechnen. Dieses hat bei Digi-View eine Auflösung von 320 mal 200 Bildpunkten, mit 32 oder 4096 Farben. Diese Auflösung entspricht zwar nur der des Commodore 64, der aber dann normalerweise nur über zwei Farben verfügt. Mit mehr Farben lassen sich jedoch feinere Abstufungen erzielen, wodurch die Auflösung subjektiv höher erscheint.

Der Nachteil der 4096 Farben im Hold-and-Modify-Modus (HAM) besteht darin, daß noch kein Zeichenprogramm diesen Modus unterstützt. Bilder mit 32 Farben dagegen verarbeitet Deluxe Paint oder Graphicraft problem-



Falschfarben-Aufnahmen haben einen besonderen Reiz



Zoomen während der Digitalisierung ergibt tolle Effekte

le Zauberwelt

los. Gerade darin liegt der Reiz der Digitalisierung. Man erhält ein sehr wirklichkeitsgetreues Bild, das man auf vielfältige Art und Weise manipulieren kann. Wem schon immer ein verschmitztes Lächeln auf einem Hundertmarkschein fehlte, der kann auf den digitalisierten Schein das Lächeln der Mona Lisa zaubern. Und wer schon immer mal J.R. als Gartenzwerg im Vorgarten haben wollte, hat keine Probleme, ein entsprechendes Bild zu montieren.

Die optischen Filter, die man zum Farb-Digitalisieren benötigt, enthält das »Digi-View«-Paket übrigens schon. Faszinierende Effekte entstehen, wenn man bei der Digitalisierung auf die Filter verzichtet und das Objekt vor jeder Aufnahme etwas verschiebt. Die entstehenden Falschfarben-Aufnahmen haben durchaus künstlerischen Wert. Wenn man à la Pop Art durch andere Farben einen Verfremdungseffekt erreichen möchte, braucht man die Bilder nur mit einem Zeichenprogramm nachzubearbeiten. Eine interessante Anwendung der Falschfarbentechnik findet sich zum Beispiel bei Multispektralaufnahmen, wie sie in der Wissenschaft verwendet werden. Auch Bildvorlagen kann man digitalisieren und weiter bearbeiten. Selbst fehlbelichtete Dias lassen sich so mit einem Zeichenprogramm reparieren und Farbnegative für die Beurteilung als Positive wiedergeben. Für professionelle oder semiprofessionelle Anwendungen aber reichen die mitgelieferten Plastikfilter wahrscheinlich nicht aus. Hier rentiert sich der Kauf von teuren Glasfiltern. Für die Heimanwendung genügen die mitgelieferten Filter aber auf jeden Fall.

Einen Haken hat die Sache aber: Die Farbbilder sind zwar von beachtlicher Qualität, die Aufnahmetechnik eignet sich jedoch nur für Digitalisierungen von unbewegten Objekten. Die Zeit, um ein komplettes Farbbild aufzunehmen, beträgt etwa 40 Sekunden mit Filterwechsel. Aufnahmen von einem Videorecorder ohne einwandfreies Standbild oder Tier-Digitalisierungen sind fast ein Ding der Unmöglichkeit. Hier erzielen Sie mit einem Foto von dem Tier oder vom Videorecorder-Standbild bessere Ergebnisse.

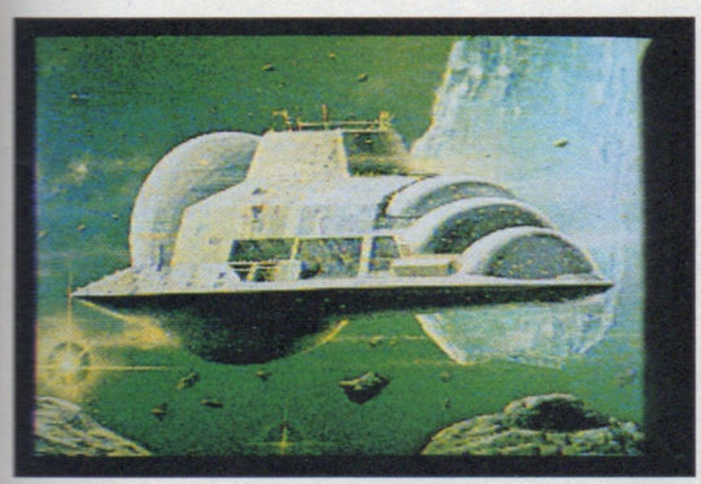
Tricks und Experimente

Ein anderer Digitalisierungs-Modus arbeitet mit der maximalen Auflösung des Amiga mit 640 mal 400 Bildpunkten (Interface) in 16 Graustufen, wodurch exzellente Schwarzweiß-Bilder entstehen. Übrigens, keine Angst vor dem berüchtigten Flimmern im Interlace-Modus: Digitalisierte Bilder sind prinzipiell so kontrastarm, daß das Bild bemerkenswert ruhig wiedergegeben wird. Den Verlust der Farbe macht die höhere Auflösung mehr als wett. Mit dieser Aufnahmetechnik können Sie ohne weiteres Ihre Freunde bei irgendwelchen Tätigkeiten digitalisieren oder ein Familienfoto aufnehmen. Eine Aufnahme dauert etwa 20 Sekunden. Dabei kann man am Bildschirm mitverfolgen, welcher Bildausschnitt schon digitalisiert wurde. Die Aufnahme erfolgt von links nach rechts, wobei sich das Bild aus 640 senkrechten Streifen (im Farbmodus sind es 320 Streifen) langsam aufbaut.

Dies eröffnet ein weites Feld für Experimente. Wenn man schnell genug ist, kann man sich zum Beispiel selbst begrüßen oder mit sich selber schachspielen. Das Prinzip ist folgendes: In der Bildmitte ist der Tisch mit dem Schachbrett zu sehen, eine Person sitzt auf dem linken Stuhl. Sobald der Schachspieler digitalisiert ist, taucht er unter dem Tisch hindurch (außerhalb des Blickwinkels der Kamera natürlich) und setzt sich auf den rechten Stuhl, während der Digitalisierer gerade über das Schachbrett streicht. Erst Sekunden später wird der rechte Stuhl mit dem Spieler digitalisiert.

Da Deluxe Paint auch das hochauflösende Bildformat unterstützt, fordert es eine weitere Manipulation geradezu heraus. Setzen Sie zum Beispiel Sprechblasen in das Bild ein oder bringen Sie durch Änderung der Farbskala Farbeffekte in das Bild. Ihrer Fantasie sind dabei keine Grenzen gesetzt. Da der hochauflösende Modus »nur« über 16 Graustufen verfügt, bürgen weiches Licht und eine gleichmäßige Beleuchtung für Qualität. Außerdem empfiehlt sich ein Stativ für die Kamera.

Der Umgang mit der mitgelieferten Software ist leicht zu erlernen, da sie sowohl mit Pull-down-Menüs arbeitet als auch auf Tastendruck reagiert. Sie errechnet aus den Daten des Digitalisier-Moduls nicht nur die Bilder, son-



4096 Farben: fast schon Fotoqualität



Verblüffend realistisch mit nur 32 Farben



Wetten, daß Sie diesen Herrn noch nie lächeln sahen? Mit »Deluxe Paint« sind auch in der höchsten Auflösung noch Bildmanipulationen möglich.

dern erlaubt auch verschiedenste Manipulationen. Das Verhältnis der drei Grundfarben zueinander, die Helligkeit, der Kontrast und die Farbsättigung sind ebenso per Software-»Schieberegler« änderbar wie die Bildschärfe. Geringere Bildschärfe bedeutet weichere Übergänge, aber weniger Details. Eine Veränderung dieser Werte erfordert übrigens keine erneuten Digitalisier-Vorgänge. Einmal digitalisiert, rechnet das Programm nur die Daten im Speicher den Einstellungen entsprechend um. Für eine Archivierung oder zur späteren Nachbearbeitung der digitalisierten Kunstwerke kann man die Bilder auf Diskette speichern. »Digi-View« erlaubt wahlweise die Speicherung eines einzelnen Farbauszugs (in Graustufen), des gesamten Bildes im Standard-Format (für Malprogramme) oder in einem »Digi-View«- eigenen Format.

Der Digitalisierer erscheint auf den ersten Blick etwas langsam, wenn man aber länger mit ihm arbeitet, merkt man schnell, daß er für Heimanwendungen ausreichend, wenn nicht sogar, dank der Trickmöglichkeiten, ideal ist. Momentan macht der Preis von 758 Mark »Digi-View« auf jeden Fall für alle interessant, die auf dieses höchst faszinierende Gebiet näher eingehen wollen. Es leistet sicherlich jedem, der Bilder oder Bildvorlagen wie Schaltpläne oder erklärende Abbildungen weiterverarbeiten möchte, gute Dienste. Selbst für kleine Labors kann es je nach Aufgabenstellung eine preiswerte Alternative sein.

(Prof. Köhler/ts)

Info: INTERPLAN, Nymphenburger Straße 134, 8000 München 19, Tel.: 089/1 234066

Bilder aus Bits und Bytes

Man muß kein Künstler sein, um mit dem ST prächtige Bilder zu schaffen. »Digitalisieren« heißt das Zauberwort, mit dem Sie jedes beliebige Motiv auf den Monitor bannen.

Malprogramme sind hervorragende Werkzeuge für technische Zeichnungen und Bilder, in denen geometrische Figuren dominieren. Motive der realen Welt aber überfordern jedes noch so aufwendige Zeichen- oder CAD-Programm. Naturgetreue Abbildungen müssen entweder mühsam Punkt für Punkt zusammengestellt werden, oder deren Eingabe von Hand ist ganz unmöglich. Doch auch hier hat die Technik Mittel und Wege gefunden. Sogenannte »Digitizer« wandeln analoge Bildsignale, wie sie zum Beispiel Videokamera oder Videorecorder liefern, in digitale Signale um, die der Computer verarbeitet. Wie genau das digitalisierte Bild der Wirklichkeit entspricht, hängt im wesentlichen von drei Faktoren ab: Erstens muß die Hardware, also Bildquelle und Digitizer, das Bild in eine genügend große Anzahl von Punkten zerlegen. Je mehr Punkte, desto feinere Strukturen sind erkennbar. Zum zweiten muß der Computer mit einer möglichst hohen Auflösung mithalten können. Ein dritter wesentlicher Punkt ist schließlich die Anzahl der verfügbaren Farben beziehungsweise Graustufen, die der Computer liefert. Je mehr Abstufungen darstellbar sind, desto mehr Tiefe und Plastizität erhält das Bild.

Für den ST bietet Print-Technik seit kurzer Zeit den »Video-Digitizer« mit beachtlichen Leistungen (siehe Bild 1) an. Er wird in zwei Ausführungen angeboten. Das Standardmodell gestattet eine Auflösung von 256 x 256 Punkten, das erweiterte Modell eine von 512 x 256 Punkten. Beide Geräte eignen sich zur Verarbeitung eines Videosignals, wie es jede Videokamera oder jeder Videorecorder liefert. Der Anschluß an den Computer erfolgt einfach über den Centronics-Port. Die Zeit für eine Digitalisierung beträgt rund fünf Sekunden, wobei das Objekt für diesen Zeitraum möglichst unbewegt verharren sollte.

Beim Videorecorder wird dies über die Standbildfunktion erreicht. Dies gestaltet sich allerdings mit den gängigen Videosystemen wie »VHS« und »Betamax« problematisch. Mit diesen Recordern erhalten Sie Standbilder leider nie ganz streifenfrei. Die beiden Heim-Videosysteme, die ein gestochen scharfes Standbild ohne zusätzlichen »Schnee« schaffen, heißen »Video 8« und »Video 2000«. In den meisten Anwendungsfällen arbeitet man jedoch am besten mit einer Videokamera.



Brauchbare Geräte, dies sind zum Beispiel Kameras für Überwachungsaufgaben, erhält man bereits ab 300 Mark im Elektronikfachhandel. Hier benötigt man unbedingt zusätzlich ein Stativ, damit die Kamera während der langen »Belichtungszeit« nicht wackelt. Die Software ist für beide Modelle des Digitizers bedienungsgleich und wird durch GEM voll unterstützt. Pull-down-Menüs erlauben den Zugriff auf alle Funktionen. Die Darstellung des Bildes hängt nun davon ab, welcher Monitor Verwendung findet. Beim Farbmonitor empfiehlt sich die niedrigste Auflösung, also 320 x 200 Pixel in 16 Farbstufen. Beim Monochrommonitor ist natürlich nur die höchste Auflösung von 640 x 400 Pixeln in einer Farbe verfügbar. Da jedoch auch hier 16 Grauabstufungen über Pixelmuster realisiert werden, fällt die effektive Auflösung nicht höher aus als beim Farbmonitor.

Nun unterscheidet sich aber die Auflösung der Hardware von der des Monitors. Um ein komplettes Bild in einer 512 x 256 Pixel umfassenden Auflösung darzustellen, muß das Bild also entsprechend gestaucht werden. Dennoch speichert der Computer alle erfaßten Daten. Die zunächst überschüssige Auflösung kommt nämlich bei der Ausschnittvergrößerung voll zur Geltung. Über den Menüpunkt »VIEW ZOOM« definieren Sie mit Hilfe der Maus einen beliebig großen rechteckigen Ausschnitt, der anschließend auf Bildschirmformat vergrößert wird. Dieser Vorgang läßt sich auch in bereits vergrößerten Ausschnitten mehrfach wiederholen. So betrachtet man Details aus der Nähe, oder beschränkt das endgültige Bild auf den wesentlichen Ausschnitt. Natürlich ist dabei schnell die Grenze erreicht, ab der jede weitere Vergrößerung nur noch gröbere Quadrate als Pixel produziert. Mit einer Videokamera umgehen Sie dieses Problem aber einfach, indem Sie das Motiv aus geringerer Entfernung digitalisieren.



Motive aller Art lassen sich mit dem Video-Digitizer bequem verarbeiten

ZOOM als Lupe

Bei Verwendung des Farbmonitors lassen sich alle 16 Farbregister beliebig verändern. Die Farbauswahl erfolgt dabei ähnlich wie bei Neochrome durch einfaches Anklicken der neuen Farbe in einem Untermenü. Ist einmal eine passende Farbbelegung für ein Bild gefunden, speichert man die ganze Palette unter einem frei wählbaren Namen auf Diskette und lädt sie bei Bedarf wieder neu. So lassen sich schnell günstige Farbabstufungen oder Verfremdungseffekte erzeugen.

Einmal erfaßte Bilder bringen Sie natürlich auch einfach zu Papier. Dazu stehen dem Programm eigene Druckertreiber zur Verfügung. Die Größe des Ausdrucks steht vom Briefmarken- bis zum DIN-A4-Format frei. Standardmäßig sind Epson- oder Itoh-kompatible Matrixdrucker vorgesehen. Bei Farbbildern steigt man auf einen Farbausdruck um, den zum Beispiel der Tintenstrahldrucker Canon PJ-1080A bietet und der sehr ansehnliche Ergebnisse erzielt.





Viele Tricks zur Nachbearbeitung der Motive bietet die »Toolbox«

Die fertigen Bilder werden auf verschiedene Weise auf Diskette gespeichert. Vorgesehen ist die Speicherung im 256x256- beziehungsweise im 512x256-Punkte-Format, Monochrom-Bilder im Doodle-Format, Farbbilder dagegen im Neochrome-Format. In beiden Fällen ermöglicht eine zusätzlich mitgelieferte Routine die Konvertierung in das Degas-Format. Dies gewährleistet die Kompatibilität zu den meistverbreiteten Malprogrammen.

Zur Nachbearbeitung der Bilder bietet Print-Technik für 178 Mark ein Programm mit der Bezeichnung »Digitizer Toolbox« an, das Bilder im Neochrome- und im Degas-Format bearbeitet. Nicht an einen bestimmten Video-Digitizer gebunden, schließt dieses Programm eine große Lücke, indem es Funktionen bietet, die man bei fast allen gängigen Zeichen-Programmen für den ST vermißt.

Aus den geladenen Bildern kann es beliebige Ausschnitte definieren, separat als Symbole kennzeichnen und speichern. Es hält zwei Bilder gleichzeitig im Speicher, zwischen denen man einfach hin- und herschaltet. Es läßt sich also ein Ausschnitt aus dem einen Bild herausnehmen und in das andere Bild einkopieren. Collagen erstellen Sie so ganz einfach und schnell. Ein gewählter Ausschnitt ist beim Einkopieren zudem noch änderbar. Ob größer oder kleiner, es geht alles nach Wunsch, und sogar das Verhältnis zwischen Breite und Höhe des Ausschnitts wählen Sie beliebig. Einen weiteren Effekt ergibt das sogenannte »Bending«. Die Kanten des rechteckigen Ausschnitts werden dabei beliebig gekrümmt, so daß das Bild konkav oder konvex dargestellt wird (Bild 2). Ein besonderer Menüpunkt ist das Drehen der Ausschnitte. Bis auf ein zehntel Grad genau läßt sich der Drehwinkel bestimmen, unter dem die Wiedergabe dann erfolgt. Man stellt diesen Winkel entweder vorher fest ein oder dreht das Bild während des Einkopierens direkt mit Hilfe der Funktionstasten.

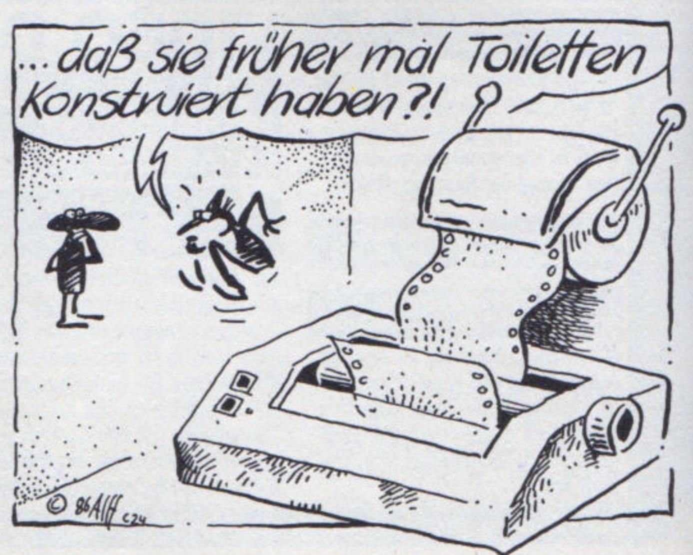
Eine endgültige Nachbearbeitung der Bilder bleibt dann den verschiedenen Malprogrammen wie Degas oder Neochrome überlassen. Damit werden beispielsweise Konturen nachgezogen, unwichtige Details gelöscht oder allgemeine Retuschen vorgenommen. Auch erklärender Text läßt sich mit diesen Programmen leicht einfügen.

Die Anwendungen, die sich aus einem solchen System ergeben, kennen kaum eine Einschränkung. Wer sich mit Computerkunst befaßt, erhält mit dem »Video-Digitizer« für 598 Mark beziehungsweise 898 Mark für die verschiedenen Modelle ein hervorragendes Hilfsmittel. Aber auch Grafiker versetzt dieses System in die Lage, beispielsweise Logos oder ähnliches schnell mit dem Computer zu erfassen und zu bearbeiten. Diashows oder Bilder für Grafikadventures sind mit der Digitalisierung schnell erzeugt. Auch Trickfilmeffekte erzielen Sie durch Mehrfachdigitalisierung oder leichte Änderungen an einem Bild ohne Schwierigkeiten. Ebenso ist die grafische Katalogisierung von Objekten denkbar.

Die Grenzen der videogestützten Digitalisierung liegen eigentlich nur im Erkennen sehr kleiner Details. So ist zum Beispiel das Digitalisieren von Schaltplänen ein Problem. Durch die feinen Linien und zu geringen Kontraste beim Digitalisieren leidet die Schärfe des Bildes. Wer jedoch den Einsatzbereich seines Computers mehr in der künstlerischen Gestaltung sieht, dem stehen mit den vorgestellten Hilfsmitteln fantastische Wege zum kreativen Umgang mit Grafik offen.

(Wolfgang Czerny/Matthias Rosin/hb)





Brandneue Bücher rund um die ATARI ST

E. Jürgensmeier
WordStar für ATARI ST
April 1986, 435 Seiten

WordStar ist ein umfangreiches und leistungsfähiges Textverarbeitungsprogramm und damit sicherlich zu Recht das meistverkaufte Programm seiner Art. Doch bedeutet dies nicht unbedingt, daß es auch einfach zu bedienen ist. Hier setzt dieses Buch an: Es macht in vorbildlicher Weise mit allen Möglichkeiten von WordStar und MailMerge vertraut und ist damit eine ideale Ergänzung zum Handbuch. Es versammelt alle Informationen für den effektiven Einsatz dieser Programme auf den ATARI-ST-Computern.
Best.-Nr. MT 90208
ISBN 3-89090-208-1
DM 49,–/sFr. 45,10/öS 382,20

In Vorbereitung:
dBASE II für ATARI ST
4. Quartal 1986, ca. 250 Seiten
Best.-Nr. MT 90206
ISBN 3-89090-206-5





P. Rosenbeck
C-Programmierung unter TOS/ATARI ST
März 1986, 376 Seiten

Erst durch das Programmieren in C kann der stolze Besitzer alle Fähigkeiten seines ATARI ST ausnutzen. Für Leser mit elementaren EDV-Vorkenntnissen gibt der Autor in diesem Buch eine gründliche und leicht lesbare Einführung in das Programmieren mit dieser wichtigen und vielseitigen Sprache. An aussagekräftigen und in allen Einzelheiten erklärten Beispielen werden auch die fortgeschrittenen Aspekte der Sprache (Dateiverwaltung, Structures, dynamische Speicherverwaltung, Rekursion) ebenso ausführlich wie die Grundlagen besprochen. Besonderes Gewicht ist auf das Programmieren auf Systemebenen gelegt (Schnittstelle zum Betriebssystem TOS, Benutzung von GEMDOS, BIOS und XBIOS), so daß der Leser in die Lage versetzt wird, auch systemnahe Programme auf seinem ATARI zu erarbeiten.
• Wagen Sie den Schritt zur Profi-Programmierung auf dem ST!
Best.-Nr. MT 90226
ISBN 3-89090-226-X
DM 52,–/sFr. 47,80/öS 405,60













R. Aumiller/D. Luda
ATARI-ST-LOGO
März 1986, 236 Seiten

LOGO vereinigt viele Vorteile anderer Programmiersprachen in sich. Es ist interaktiv, listen- und prozedurorientiert, erweiterbar, einfach zu erlernen und doch komplexen Problemen gewachsen. Dieses Buch ist für Anfänger und Fortgeschrittene gleichermaßen geeignet. Bildschirmfotos, viele ausführliche Beispiele – teilweise mit Übungsaufgaben zur Vertiefung des Gelesenen – tragen zu einer guten Verständlichkeit und einem sicheren Lernerfolg bei. Doch auch der erfahrene Programmierer kommt auf seine Kosten, professionelle Anwendungen und ein Kapitel über künstliche Intelligenz runden das Spektrum ab.
Best.-Nr. MT 90223
ISBN 3-89090-223-5
DM 49,–/sFr. 45,10/öS 382,20

P. Rosenbeck
C-Programmierung unter GEM/ATARI ST
4. Quartal 1986, ca. 300 S.

GEM, die Benutzeroberfläche der ATARI-ST-Computer, gilt als außerordentlich bedienerfreundlich. Sie vereinigt herausragende grafische Darstellung und selbsterklärende, symbolische Benutzerführung. Natürlich verbirgt sich hinter dieser freundlichen Oberfläche eine außerordentlich komplexe interne Struktur.
• Das Buch zeigt, wie man mit der Programmiersprache C die interessantesten Merkmale der GEM-Benutzeroberfläche (Windows, Pull-Down-Menüs, Maus) auch in der eigenen Programmierung verwenden kann.
Best.-Nr. MT 90203
ISBN 3-89090-203-0
DM 58,–/sFr. 53,40/öS 452,40

I. Lüke/P. Lüke
Das Systemhandbuch zum ATARI ST
4. Quartal 1986, ca. 300 S.

Zwei Themen bilden die Schwerpunkte des vorliegenden Buches: Die Struktur der 68000-CPU und der ATARI 520/260 ST. Auf dieser theoretischen Basis stellen die Autoren die Programmierumgebung des ATARI 520/260 ST anhand vieler Beispielprogramme dar. Besondere Aufmerksamkeit wird der Einbindung von Maschinensprachmodulen in das Betriebssystem und in höhere Programmiersprachen (z.B. BASIC und C) gewidmet. Die Besprechung eines 68000-Assemblers und einige gerätespezifische Maschinensprachmodule runden das Buch ab.
Best.-Nr. MT 90216
ISBN 3-89090-216-2
DM 52,–/sFr. 47,80/öS 405,60

W. F. Fastenrath
ATARI-ST-BASIC-Handbuch
März 1986, 264 Seiten

Das BASIC für die ATARI-ST-Computer ist außerordentlich umfangreich und mächtig. Über 130 Befehle stehen bereit, um auch komplexere Aufgaben mit diesen Computern zu bewältigen. Die neuartige Benutzeroberfläche der Rechner erforderte ein entsprechendes »Tuning« dieser altgedienten Programmiersprache.
Dieses Buch stellt eine Anleitung zur Anwendung von BASIC auf die Erfordernisse und Möglichkeiten dieses speziellen Systems dar. Eine übersichtliche Zusammenstellung des gesamten Befehlsvorrats macht dieses Buch zu einem Hilfsbuch bei der täglichen Programmierarbeit.
Best.-Nr. MT 90205
ISBN 3-89090-205-7
DM 52,–/sFr. 47,80/öS 405,60

I. Lüke/P. Lüke
Der ATARI 520 ST
2. überarbeitete und erweiterte Auflage März 1986, 198 Seiten

Dieses Buch enthält alle Informationen, die für stolze Besitzer eines ATARI 520/260 ST wichtig sind. Die jetzt vorliegende überarbeitete und erweiterte Auflage trägt den neuesten Entwicklungen bei ATARI Rechnung. Unter anderem wurden das inzwischen deutschsprachige Betriebssystem und einige geänderte Systemausstattungsmerkmale berücksichtigt.
Best.-Nr. MT 90229
ISBN 3-89090-229-4
DM 49,–/sFr. 45,10/öS 382,20

J. Steiner/G. Steiner
GEM für den ATARI 520 ST
2. überarbeitete und erweiterte Auflage Februar 1986, 334 Seiten

Dieses Buch ist eine Einweisung in alles, was GEM für den Benutzer interessant macht. Besonders interessant für den fortgeschrittenen Anwender, aber auch für den, der »nur« die Struktur eines so komplexen Betriebssystems kennenlernen möchte, sind die Kapitel über den internen Aufbau von GEM mit seinen grafischen Merkmalen.
Best.-Nr. MT 90230
ISBN 3-89090-230-8
DM 52,–/sFr. 47,80/öS 405,60



Bestellungen im Ausland bitte an den Buchhandel oder an untenstehende Adressen.
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, ☎ 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, 1091 Wien, ☎ 0222/48 15 38-0

Irrtümer und Änderungen vorbehalten.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München



Eins plus eins gibt eins

Was bisher nur Fernsehstudios und teuren Spielhallen-Automaten vorbehalten war, damit kann sich nun auch der Amiga brüsten: Für ihn gibt es jetzt ein »Genlock«-Interface, das Videobilder vom Recorder oder von einer Kamera mit Computergrafik mischt.

Fernsehstudios setzen die Technik des Bildmischens besonders häufig ein. Hierzu einige Beispiele: Bei den meisten Nachrichtensendungen ist eine Kamera auf den Sprecher und seinen Schreibtisch gerichtet, eine andere Kamera nimmt die Hintergrundgrafik (Bilder, Titel der Sendung) auf. Ein relativ teurer Bildmischer holt sich nun das Videosignal mit dem Nachrichtensprecher und ersetzt die Hintergrundfarbe durch die Bilder und Grafiken, welche die zweite Kamera bereitstellt.

Fast schon selbstverständlich ist die Überlagerung von Bildsignalen beim Vor- und Nachspann eines Fernsehfilms, wenn alle Mitwirkenden genannt werden. Kennen Sie die Zeichen der Fernsehanstalten, die gelegentlich in der linken oberen Bildschirmecke zu sehen sind? Auch das ist Bildmischung.

Das Mischen zweier Tonsignale bereitet keinerlei Schwierigkeiten: Es genügt, einfach beide Leitungen miteinander zu verbinden. Bei Videosignalen liegt der Fall etwas anders, da diese neben der Bildinformation zusätzlich Steuersignale enthalten. Das Vertikal-Synchron-Signal zum Beispiel, das bestimmt, wann ein Bild beginnt. Der Horizontal-Synchron-Impuls wiederum legt den Start einer neuen Zeile fest. Mischt man mehrere Synchronsignale, so erhält man als Ergebnis ein verzerrtes, durchlaufendes Bild.

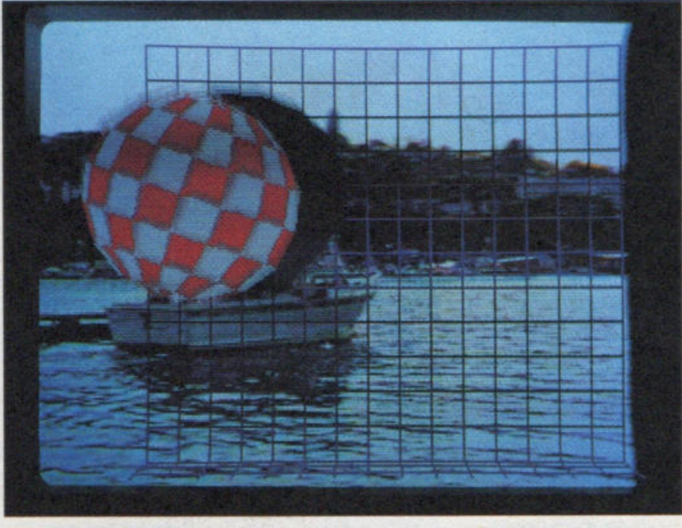
Die Lösung: Eine der beiden Videoquellen muß sich an die Steuersignale des anderen Signals anpassen. Die Amiga-Hardware ist erfreulicherweise für den Anschluß eines externen Video-Taktgebers schon vorbereitet. Das Genlock-Interface steuert den Amiga mit den Synchronimpulsen des externen Videosignals und kann sodann beide Bilder überlagern.

Das Genlock-Interface schiebt man einfach von hinten unter den Amiga, womit es auch schon mit dem Monitor-Anschluß verbunden ist, der auch alle notwendigen Spannungs- und Steuerleitungen zur Verfügung stellt. Das Monitor-Kabel steckt in einer Buchse am Interface.

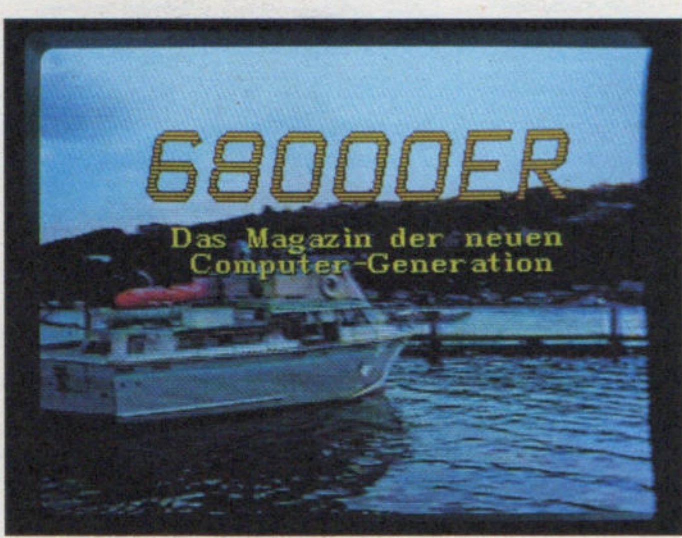
Solange keine Videogeräte am Genlock-Interface hängen, verhält sich der Amiga wie gewohnt. Sobald ein Videosignal am Videoeingang des Interface anliegt, mischt sich dieses mit dem Amigabild. Mischen bedeutet: Das Videobild ersetzt die Hintergrundfarbe der Amiga-Grafik, alle anderen Farben sind zu sehen. Angenommen, Sie haben eine Kamera an dem Videoeingang des Genlock-Interfaces angeschlossen, die eine Blume aufnimmt. Mit einem Malprogramm können Sie jetzt die Blume am Bildschirm genauestens nachzeichnen, da Sie die Blume und das Gemalte gleichzeitig vor sich auf dem Bildschirm sehen.

Über einen Schalter am Genlock-Interface wählt man zwischen drei verschiedenen Anzeigearten. Entweder ist nur die Amiga-Grafik zu sehen, oder das Videobild alleine oder das Mischbild. Ein Beispiel: Sie verbinden einen Fernseh-Tuner mit dem Video-Eingang und arbeiten mit einer Textverarbeitung auf dem Amiga. Hinter Ihrem Text sehen Sie die Nachrichten und hören den Sprecher, während Sie einen Brief schreiben. Wollen Sie jetzt eine interessante Meldung komplett auf den Bildschirm holen, genügt ein Umlegen des Schalters, und die Computergrafik blendet sich aus.

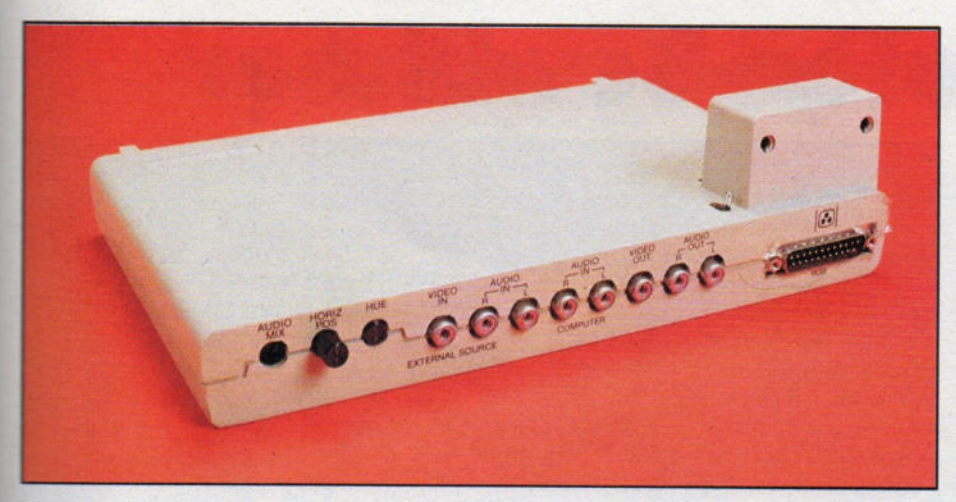
Neben dem RGB-Ausgang befindet sich am Genlock-Interface eine



Interessant für Videoamateure: Animierte Grafik, hier der Amiga-Ball, zu überlagern, ist ebenso leicht wie...



...das Einblenden von Texten, Titeln und Grafiken in den laufenden Videofilm für außergewöhnliche Effekte



Die Anschlüsse des Genlock-Interfaces lassen keine Wünsche offen

Buchse, die ein FBAS-Video-Signal für Videomonitore oder Videorecorder liefert. Hier bietet sich ein weites Betätigungsfeld für Videoamateure, ihre Videos mit Titeln, Diagrammen, Computergrafiken oder Animationen zu bereichern. Für solche Anwendungen eignet sich bestens ein Animationsprogramm wie »Deluxe Video«, das sich sowohl auf animierte Grafik, bewegte Texte als auch auf Balken- und Tortendiagramme versteht.

Doch auch die Tonseite kommt beim Genlock-Interface nicht zu kurz. Es verfügt über zwei Stereo-Eingänge für den Computer und eine weitere Tonquelle (Videorecorder, Stereo-Anlage) sowie über einen Stereo-Ausgang, an dem das gemischte Signal der beiden Tonquellen anliegt. Ein Drehregler bestimmt das Lautstärke-Verhältnis von Computersound und externem Tonsignal. Auch für das Videosignal sind zwei Regler zur Einstellung von Helligkeit und horizontaler Position vorhanden, um einen optimalen Überblendeffekt zu erreichen.

Das Genlock-Interface arbeitet leider zur Zeit nur in der amerikanischen NTSC-Norm. Der sinnvolle Einsatz in unseren Landen setzt eine Anpassung an die PAL-Norm aber unbedingt voraus. Leider stehen weder Preis noch Erscheinungstermin endgültig fest. Jeder kreative Amiga-Besitzer, der sich für das faszinierende Gebiet der Videotechnik interessiert, würde ein PAL-Genlock-Interface als große Bereicherung zu schätzen wissen. (ts)

Info: Commodore Büromaschinen GmbH, Lyoner Straße 38, 6000 Frankfurt/M. 71

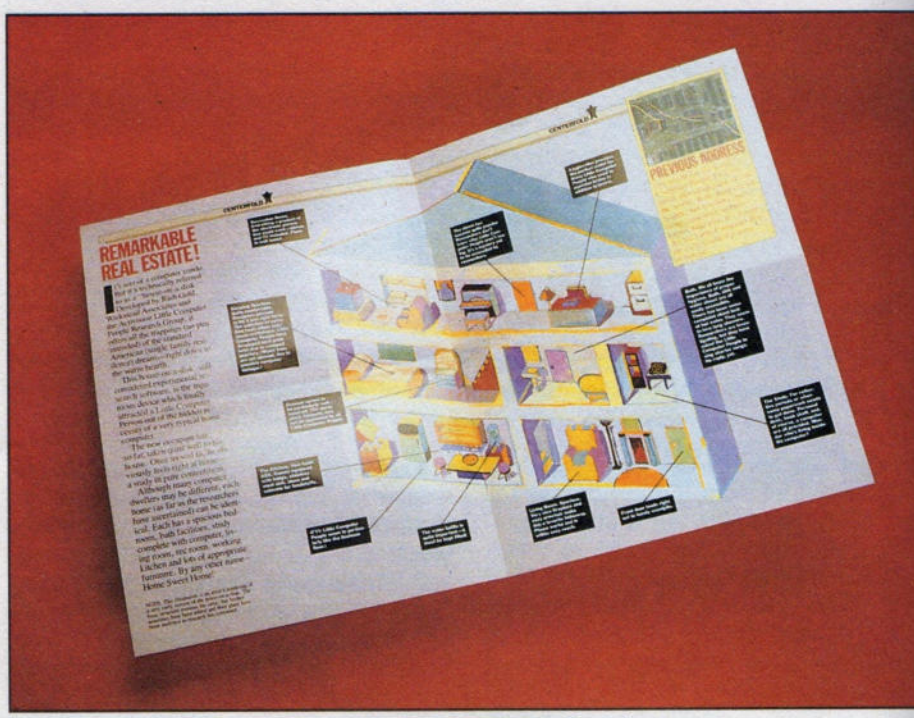


Computer-Bewohner

Nun ist es wissenschaftlich erwiesen: Auch in 68000-Computern leben kleine Männchen! Nehmen Sie an einem Forschungsprojekt teil, um die »Little Computer People« zu entdecken.

Kennen Sie schon unseren Freund Whitney? Er ist ein sehr liebenswerter Zeitgenosse und schreibt uns in seinen Briefen, daß es ihm und seinem Hund bei uns sehr gut gefällt. Seine große Leidenschaft ist Musik. Er setzt sich gerne mal ans Klavier oder legt eine Schallplatte auf. Wie wir Whitney kennengelernt haben? Tja, eines schönen Morgens schaltete ein ahnungsloser Redakteur einen Atari ST an und machte eine verblüffende Entdeckung.

Dank eines speziellen Programms namens »The Little Computer People Discovery Kit« konnten wir das Innenleben unseres Computers genauer beobachten. Denn zwischen Grafik-Chip und RAM-Bausteinen findet man ein kleines Häuschen! Doch es kommt noch besser: Es erschien ein kleines Männchen, das von Zimmer zu Zimmer marschierte. Nach einer Weile verschwand der Besucher, doch schon ein paar Minuten später tauchte er wieder mit einem Karton auf, in dem sich seine Habseligkeiten befanden: Er zog ein!



Ein Auszug aus der Zeitschrift »Modern Computer People«



Der kleine Computer-Bewohner in seinem Haus

Solche kleinen Computer-Bewohner sind ein Phänomen, das seit knapp einem Jahr bei 8-Bit-Computern zu beobachten ist. Der Erfinder der kleinen Computer-Persönlichkeiten ist kein geringerer als Amerikas Star-Programmierer David Crane.

Das »Little Computer People Project« ist ein höchst ungewöhnliches Programm. Man könnte es am ehesten als humorvolle Simulation bezeichnen, denn hier geht es nicht um Abschießen oder Punkte sammeln. Ein festes Spielziel ist auch nicht vorgegeben. Man beobachtet seinen kleinen Computer-Freund und dessen Angewohnheiten. Durch Tastatureingaben (unser Männchen versteht aber nur Englisch) können Sie versuchen, den Computer-Bewohner zu bestimmten Dingen zu animieren; zum Beispiel eine Runde Black Jack zu spielen oder seinen Hund zu füttern.

Dank der Control-Taste kann man auch für seinen Computer-Bewohner sorgen und ihm Schallplatten, Bücher und Nahrungsmittel vor die Haustür stellen. Er freut sich auch über einen Anruf und – nicht zu vergessen – über ein paar Streicheleinheiten. Auch kleine Computer-Leute brauchen Liebe!

Solche Informationen findet man in der nicht ganz ernstzunehmenden wissenschaftlichen Zeitschrift »Modern Computer People«, die der Original-Packung beiliegt. Ein hinreißendes Magazin, das auf zwölf DIN-A4-Seiten Informationen, Hintergründe und Interviews zum heißen Thema »Kleine Computer-Bewohner« bietet.

Zwei Kritikpunkte seien dennoch erlaubt: Die Grafik der getesteten Atari-ST-Version fällt zwar etwas farbenprächtiger aus als beim C64-Original, aber inhaltlich gibt es kaum Verbesserungen. Angesichts der Tatsache, daß die Atari-Version neben einem Farbmonitor 512 KByte RAM benötigt (Besitzer eines 260 ST können das Programm nur mit ROMs laufen lassen), hätte man etwas mehr zusätzliche Extras erwartet.

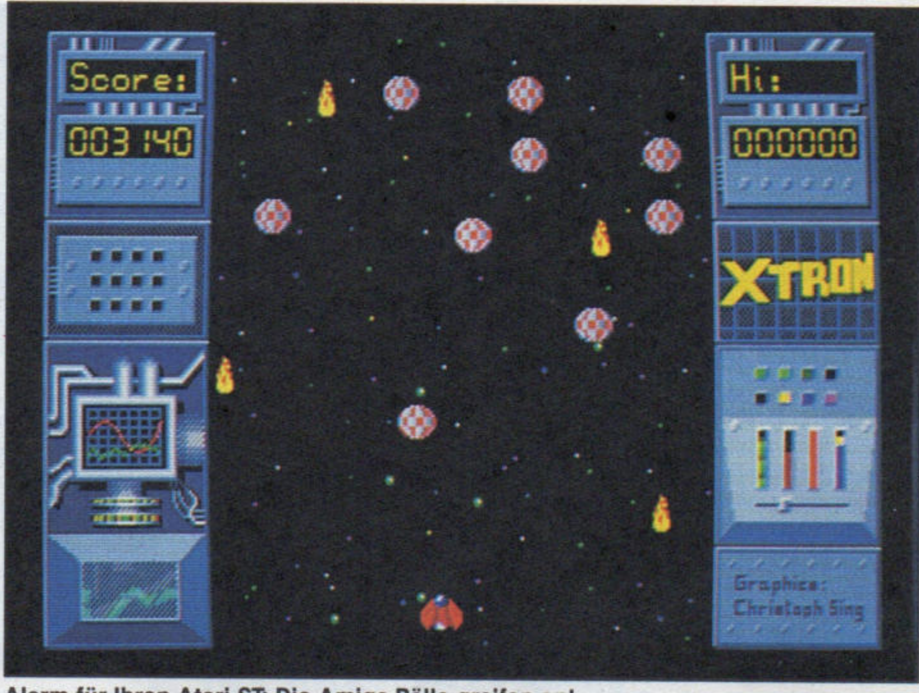
Zum anderen ist es zweifelsohne Geschmackssache, ob man sich auf Dauer an den Beobachtungen einer kleinen Computer-Person erfreut. Nach einer gewissen Zeit tut sich nämlich auf dem Bildschirm nicht mehr viel Neues. Was soll's – das »Little Computer People Project« lebt ohnehin von seiner ungeheuren Originalität, denn wer möchte seinem Freund nicht mal das kleine Männchen im heimischen Computer zeigen? (hl)

Erhältlich auf Diskette für Atari ST und Amiga (je 79 Mark). Nähere Informationen bei Activision Deutschland, Postfach 76 06 80, 2000 Hamburg 76.



Action hoch drei

Wenig Handlung, aber viel Action mit ausgeklügelten Effekten: »Xtron« verwandelt den Atari ST in einen Spielautomaten à la »Galaxians« & Co.



Alarm für Ihren Atari ST: Die Amiga-Bälle greifen an!

Mit »Xtron« kommen Sie nicht nur in den Genuß eines gepflegten Spielchens, Ihr Atari ST sorgt auch für Super-Sound. Wenn das Titelbild erscheint, ertönt nämlich ein heißer Disco-Mix aus dem Monitor-Lautsprecher. Die Programmierer haben ein komplettes Musikstück digitalisiert! Soviel Aufwand hat seinen Preis: »Xtron« schluckt satte 700 KByte.

Vor lauter schwungvoller Musik sollte man das eigentliche Spiel natürlich nicht vergessen. »Xtron« ist fast schon ein Relikt vergangener Tage. Es ist nämlich nicht mehr und nicht weniger als ein saft- und kraftvolles Ballerspiel in bester Arkade-Manier und ein naher Verwandter solcher Klassiker wie »Space Invaders«, »Galaga« und »Galaxians«.

Mit einem 08/15-Schießspiel lockt man natürlich keinen Atari mehr hinter der Hard Disk hervor. Bei »Xtron« sorgt die technische Brillanz für Spiellaune. Sie steuern mit dem Joystick Ihr Raumschiff, das am unteren Bildrand nach links und rechts bewegt werden kann. Mit dem Feuerknopf läßt man ultrahocherhitzte Lasersalven auf die Eindringlinge los. Der Aufmarsch dieser Gegner auf dem Monitor ist eine regelrechte Sprite-Invasion. Eine speziell für »Xtron« entwickelte Sprite-Routine macht diese Bildschirm-Action möglich. Knapp zwei Dutzend feindliche Raumschiffe kommen gleichzeitig herangebraust und formieren sich zum Gefecht.



Level 21: Ein gewaltiger Gegner taucht auf

Es bleibt natürlich nicht bei friedlichen Flugkünsten. Da fallen ultradynamische Mega-Schwupp-Bomben, daß es nur so eine Freude ist. Immer wieder lösen sich einige Schiffe aus dem Verband und kommen auf Ihr Raumschiff zugeschwirrt.

Es gibt 20 verschiedene Levels. Die unterschiedlichsten Feinde greifen mit ständig wechselnden Taktiken an. Daß »Xtron« sich nicht allzu ernst nimmt, zeigt sich im 7. Level ganz eindeutig. Hier nehmen die gegnerischen Raumschiffe die Form des berühmt-berüchtigten rot-weißen Amiga-Balls an.

Jenseits des 20. Levels wartet dann eine gewaltige Herausforderung: ein unglaublicher Gegner, fast unmöglich zu schlagen. Unser Testpilot konnte sich noch nicht bis zu dieser Grenze vorkämpfen, doch nach jüngsten Gerüchten lauert dort ein ganz dicker Otto; quasi der Boß aller bisherigen Raumschiffe. Wer auch diesen Brocken beseitigt, darf sich noch einmal mit allen anderen 20 Levels herumärgern – allerdings wird es jetzt ein wenig schwieriger: Tödliche Galakto-Doppelpuff-Bomben schwirren jetzt über die Mattscheibe.

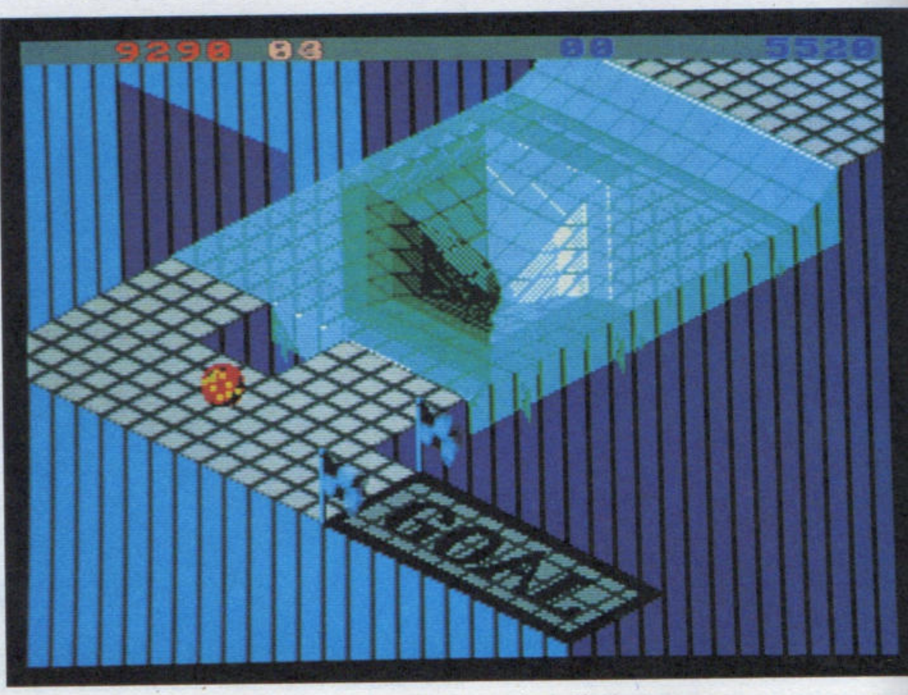
Elegante Sprite-Animation und butterweiches Scrolling sind die besonderen Kennzeichen des schnellen Actionspiels. Es läuft nur mit einem Farbmonitor und verlangt auch ordentlich RAM: Man benötigt entweder einen MByte- oder einen 512-KByte-ST mit installierten ROMs. Bei der 512-KByte-Version muß man allerdings auf die spektakuläre digitalisierte Musik verzichten. Die Präsentation des Programms läßt leider zu wünschen übrig. Verpackung und Dokumentation erhalten das Prädikat »armselig«.

Wenn Sie von hochintelligenten Adventures, RAM-Disks und C-Programmierung mal etwas Erholung suchen und nicht gerade ein Feind altmodischer, kerniger Actionspiele sind, haben Sie an »Xtron« Ihre helle Freude. Sonderlich geistreich und innovativ ist das Programm zugegebenerweise nicht gerade, aber ausgesprochen unterhaltsam. (hl)

Erhältlich auf Diskette für Atari ST (79 Mark). Nähere Informationen bei RDS-Software, Jakobstr. 8a, 6096 Raunheim

Auf die Murmel gekommen

Ein Superlativ gefällig? Auf dieser Seite stellen wir Ihnen das unserer Meinung nach beste Geschicklichkeitsspiel vor, das es momentan für den Amiga gibt.

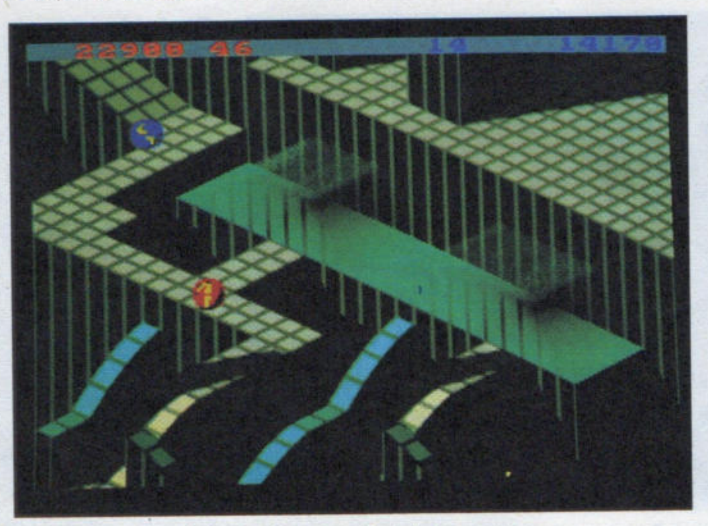


Vorsicht, Eis! Ganz sachte schlittert die Kugel dem Ziel entgegen

Wer sich unlängst den Traumcomputer Amiga zugelegt hat und zu den Spiele-Fans zählt, der weiß bald, was Frust bedeutet. Das Software-Angebot hält sich zur Zeit noch in Grenzen, was ja auch verständlich ist. Aber leider hapert's auch bei der Qualität oft gewaltig: Selbst die Profi-Programmierer brauchen halt ein ganzes Weilchen, um einen neuen Computer richtig auszureizen.

Doch jetzt ist »das« Programm erschienen, auf das Amiga-Spieler schon seit Monaten fieberhaft gewartet haben. Electronic Arts hat eine fantastisch gute Adaption des Spielautomaten »Marble. Madness« herausgebracht.

Der Automat kam im Frühling 1985 in die Spielhallen und avancierte aus dem Stand zum Oberknüller. Auch in den Büros von Electronic Arts wurden die Mitarbeiter vom »Marble Madness«-Fieber befallen, was dann letztendlich auch zu der Idee führte, die Heimcomputer-Umsetzungen für dieses Spiel in Angriff zu nehmen.



Jeder Level fasziniert mit neuen Spielelementen

Ein oder zwei Spieler (die gleichzeitig antreten können) steuern je eine Murmel, die man in sechs verschiedenen Spielstufen in einer Art Hindernislauf vom Start bis zum Ziel steuern muß. Jeder Level ist etwa vier bis zwölf Bildschirme lang und scrollt rauf und runter, wenn die Murmel ihres Weges rollt. Schafft ein Joystick-Künstler alle Strecken, geht es wieder von vorne los. Aber natürlich wird es dann viel schwieriger, und neue Schikanen tauchen auf.

Es gibt keine bestimmte Anzahl von Leben, sondern ein Zeitlimit, innerhalb dessen man das Streckenende erreichen muß. Wenn man einen Level beendet und noch ein paar Sekunden übrig hat, wird diese Restzeit zum Limit des nächsten Spielabschnitts dazuaddiert. Der erste Level - fast ein »Probedurchlauf« - hat eine Sonderstellung. Hier wird die restliche Zeit am Ende nicht mit in die nächste Runde genommen.

Ab dem zweiten Level geht's dann rund: Das Spiel ist ein Sammelsurium von aberwitzigen Monstertypen (Murmel-Mampfer, Säure-Schleim, Super-Staubsauger, Mörder-Murmeln) und halsbrecherischen Streckenabschnitten. Die exzellente 3D-Grafik hat Automaten-Qualität.

Die Musik kommt mit der Grafik nicht ganz mit, denn im Gegensatz zum sechsstimmigen Automaten hat der Amiga »nur« vier Stimmen zur Verfügung. In einigen Levels gelang die Musikuntermalung trotzdem sehr gut. Vor allem mit zwei externen Stereo-Boxen wird »Marble Madness« auch zum Sounderlebnis.

Besonders zu zweit hat das Programm einen ungeheuren Unterhaltungswert - vor allem dann, wenn die beiden Spieler nach der Brutalo-Methode vorgehen und sich gegenseitig herunterschubsen wollen. Es ist nur schade, daß auf die sonst obligatorische High-Score-Liste verzichtet wurde. Da das Spiel nicht vollständig in Assembler geschrieben ist, beansprucht es extrem viel Speicherplatz: Obwohl jeder Level einzeln nachgeladen wird, benötigt das Programm 512 KByte Arbeitsspeicher.

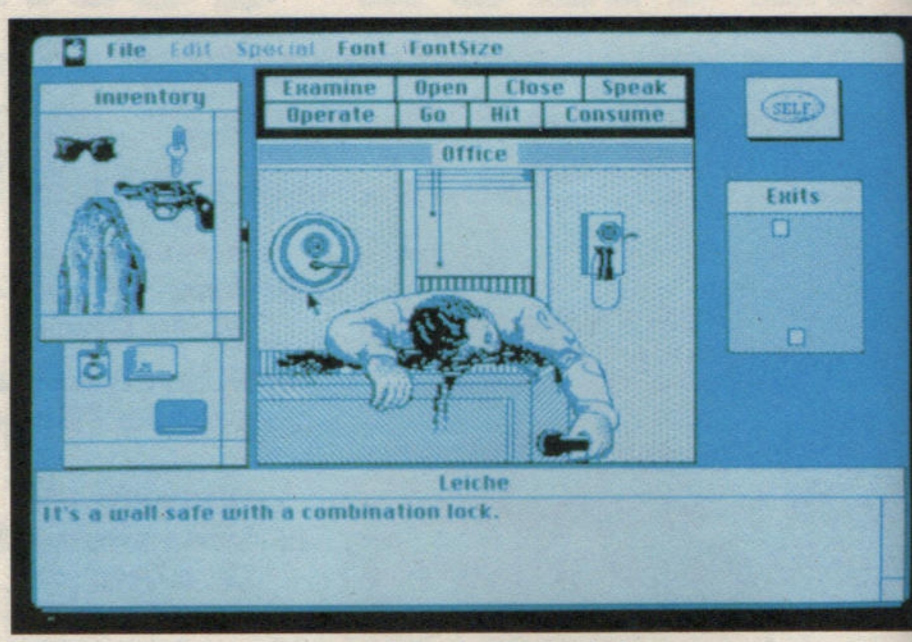
Für Besitzer eines Macintosh oder Atari ST gibt es schlechte Nachrichten, denn für diese beiden Computer sind vorerst keine Umsetzungen geplant. Verspielte Amiga-Fans haben hingegen reichlich Grund zur Freude: »Marble Madness« nutzt ihren Computer zwar nicht 100prozentig aus, aber das erstklassige Geschicklichkeitsspiel läßt alle derzeitigen Konkurrenzprodukte vor Neid erblassen. (hl)

Erhältlich auf Diskette für Amiga 512 KByte (99 Mark). Nähere Informationen bei Ariolasoft, Carl-Bertelsmann-Str. 161, 4830 Gütersloh, Tel. (05241) 8053-0.



Die Kugel im Revolverlauf

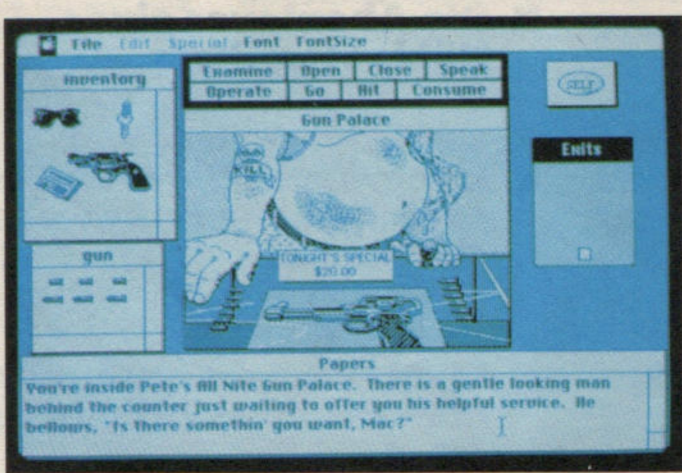
Krimi-Adventures sind derzeit groß in Mode. Das bislang beste kommt aus den USA: »Deja Vu« hat nicht nur eine gewitzte Handlung mit starken Grafiken; die Benutzerführung zählt auch zur ersten Garnitur.



Eine Leiche – und alle glauben, daß Sie der Mörder sind

Abenteuerspiele sind eine recht schöne Sache und erfreuen sich anhaltender Beliebtheit. Leider hat das Spielprinzip einen Nachteil: Der Spieler wird sich immer damit herumärgern, daß das Programm nur einen bestimmten Wortschatz hat. Nun geht das beliebte Spielchen »Wie sag ich's meinem Computer?« los und man tippt Wort für Wort in der verzweifelten Hoffnung ein, daß dem geneigten Adventure jene Vokabel auch geläufig ist. Solch ein Unterfangen mausert sich mitunter zur regelrechten Software-Tragödie.

Ein amerikanisches Programmierteam namens Icom Simulations ließ sich nun etwas ganz Neues einfallen. Mit »Deja Vu« wurde das erste Abenteuerspiel entwickelt, das vollen Gebrauch von der mausorientierten Benutzerführung von Amiga, Mac & Co. macht.



Einkaufsbummel im »Gun Palace«

Sie klicken einfach eines von acht Verben aus einer Menüleiste und anschließend den Gegenstand im Bild an, auf den sich die Handlung auswirken soll. Das klingt zwar recht simpel, aber man ist damit keinesfalls in seinen Taten eingeschränkt. Durch sinnvolle, vielseitige Kommandos wie zum Beispiel »Operate« erspart man sich eine Menge artverwandter Fuzzi-Vokabeln. Dazu gibt es Windows in verschwenderischer Menge. Neben einem Text- und einem Grafik-Fenster zeigen separate Windows die Dinge, die Sie gerade bei sich tragen und alle vorhandenen Ausgänge. Wenn man einen Gegenstand wie zum Beispiel eine Schreibtisch-Schublade öffnet, erscheint prompt ein neues Fenster, in dem alle Dinge gezeigt werden, die sich in der Schublade befinden.

Die Story ist auch nicht von Pappe: Sie erwachen mit fürchterlichen Kopfschmerzen aus der Bewußtlosigkeit und können sich an überhaupt nichts mehr erinnern – Sie haben Ihr Gedächtnis verloren. Schnappen Sie sich Trenchcoat und Revolver, die an der Wand hängen und erforschen Sie das Gebäude. In einem Zimmer stolpern Sie über eine Leiche und damit beginnt ein regelrechtes Kesseltreiben. Die Polizei ist nämlich hinter Ihnen her, und auf den Straßen lauern alle möglichen Typen, die es auf Ihr Leben abgesehen haben.

»Deja Vu« ist nicht nur sehr spannend, sondern auch angenehm selbstironisch und humorvoll. Nachdem ein Halunke Sie gerade mit einer Kugel erwischt hat, erscheint die launige Bemerkung: »Sie schämen sich furchtbar, weil Sie nicht in frischen Socken sterben.« Und dann gibt es noch die Beschreibung eines Stuhls, der offensichtlich zum Verhören von Gefangenen dient: »Der Stuhl hat Fesseln, um Arme und Beine festzubinden. Vielleicht ist es der Behandlungsstuhl eines Zahnarztes.«

»Deja Vu« gehört mit zum Besten, was an Abenteuerspielen auf dem Markt ist. Die ungewöhnliche Benutzerführung begeistert. Sie ist nicht nur für Einsteiger sehr gut geeignet, sondern bietet auch alten Adventure-Hasen ein ganz neues, ausgesprochen angenehmes Spielgefühl.

Wie Sie an unseren Schwarzweiß-Fotos sehen können, testeten wir die Macintosh-Version. Eine Amiga-Adaption soll bei Erscheinen dieser Ausgabe bereits fertig und die Umsetzung für den Atari ST dürfte nur noch eine Frage der Zeit sein.

Vom selben Programmiererteam gibt es bereits ein brandneues Adventure, das nur auf dem 512 KByte-Macintosh läuft und eine weitere Steigerung des Spielvergnügens verspricht: »Uninvited« bietet den gleichen Spielkomfort neben gepflegter Grusel-Handlung und digitalisierten Soundeffekten. Mehr darüber demnächst in unserem Stammheft Happy-Computer. (hl)

Erhältlich auf Diskette für Macintosh 128 KByte und Amiga 256 KByte (zirka 100 Mark). Nähere Informationen bei Rushware, An der Gümpgesbrücke 24, 4044 Kaarst 2.



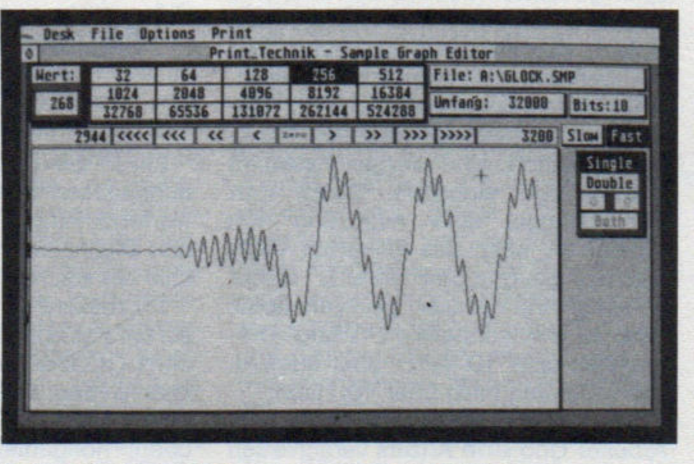
Soundmaschine

Ein Computer mit schnellem Prozessor und viel Speicherplatz ist zum Verfremden von Tönen wie geschaffen. Ob Sprachausgabe oder Musikuntermalung, da ist alles drin.

Nimmt man Musik mit dem Kassettenrecorder auf, so werden analoge Schwingungen auf dem Magnetband der Kassette gespeichert. Verwendet man aber anstelle des Recorders einen Computer, muß man die analogen Schwingungen in digitale Werte umwandeln. Diese Werte sind Reihen von Bits, die der Computer als Zahlenwerte verarbeiten kann. Die Umwandlung übernimmt ein sogenannter Analog/Digital-Wandler. Wie genau die digitalen Werte nun die ursprünglichen analogen Signale beschreiben, hängt von zwei Faktoren ab. Zum einen ist die Zahl der Amplitudenmessungen, also die Abtastrate pro Sekunde entscheidend. Gehen wir von einer Sinusschwingung von einer Sekunde Dauer aus. Mißt man die Höhe dieser Schwingung nun fünfmal pro Sekunde, spiegeln die Werte die ursprüngliche Kurve nur sehr grob wider. Verwandelt man die Werte mit einem Digital/Analog-Wandler wieder in eine Schwingung zurück, so erhält man statt einer glatten Kurve ein eckiges, treppenförmiges Gebilde. Je mehr Messungen man pro Sekunde ausführt, desto genauer entspricht die rekonstruierte Schwingung dem Original. Der zweite Punkt ist die Meßgenauigkeit. Die Anzahl der Bits pro Meßwert bestimmt die Genauigkeit. Bei 8 Bit kann man einen analogen Wert in einen Bereich von 256 Stufen unterteilen. Bei 10 Bit steigert sich die Genauigkeit bereits um das Vierfache. Man erhält einen Meßbereich zwischen 0 und 1023. Je besser die Klangqualität beim Abspielen der digitalisierten Aufnahme sein soll, desto mehr Speicherplatz benötigt man zwangsläufig. Mit 30 000 Byte Speicherplatz läßt sich beispielsweise ein Ton von einer Sekunde Dauer bei 8 Bit und einer Abtastrate von 30 000 Hz speichern. Das setzt nun schon einen Computer mit so viel Speicherplatz wie den Atari ST voraus, um längere Passagen in akzeptabler Qualität wiederzugeben.

Print Technik bietet einen Sampler mit der Bezeichnung »Sound Master Pro« an. Die Hardware befindet sich in einem Kästchen, das man einfach über ein Kabel mit dem ROM-Port des ST verbindet. Neben der Eingangsempfindlichkeit kann man den Klang bei der Ein- und Ausgabe separat regeln.

Klänge aufnehmen, wiedergeben und auf Diskette speichern, gehört zum Repertoire des im Lieferumfang enthaltenen Programms. Die Abtastrate läßt sich zwischen 5 und 90 kHz in 1-kHz-Schritten einstellen. Als Spitzenwert gibt Print Technik 90 000 Analog/Digital-Wandlungen pro Sekunde an. Der Sampler, der auf eine maximale Genauigkeit von 10 Bit ausgelegt ist, erreicht so Hi-Fi-Klangqualität. Für diese hohe Klangqualität benötigt das Programm pro Messung zwei Byte. Pro Sekunde Aufzeichnung in höchster Klanggüte verschlingt das Programm 180 000 Byte. Aus Speicherplatzgründen begnügt man sich also in der Regel entweder mit geringeren Abtastraten oder setzt, für die Aufnahme von längeren Passagen, die Genauigkeit auf 8 Bit herab. Dadurch verringert sich der Speicherbedarf um die Hälfte.



Der Ton im Bild mit dem »Sample Graph Editor«

Das Programm prüft, wieviel RAM zur Verfügung stehen und nutzt den Platz voll aus. Je höher die RAM-Ausstattung des ST, desto längere Passagen kann man aufzeichnen. So schafft der Atari ST+ mit einem Megabyte Speicherplatz bei einer Abtastrate von 10 kHz und einem Meßbereich von 8 Bit immerhin rund 80 Sekunden Aufnahme.

Vergrößert oder verkleinert man bei der Wiedergabe die Abtastrate, so verändert sich auch die Abspielgeschwindigkeit des Samples. Dies ruft dann Verfremdungseffekte hervor. Den Speicherbereich, auf den sich die Aufnahme oder Wiedergabe bezieht, kann man genau festlegen. So lassen sich einzelne Worte oder Klangpassagen ausgeben oder in eine neue Reihenfolge bringen. Die Klänge kann man auch über die Tastatur in verschiedenen Tonhöhen einzeln »spielen«. Auf Wunsch zeichnet das Programm sogar eine Klaviatur auf den Bildschirm, die über die Maus bedient wird.

Normalerweise erfolgt die Ausgabe der gesampelten Klänge über einen Verstärker oder Kopfhörer. Doch auch der interne Lautsprecher des Monitors eignet sich zur Wiedergabe. In diesem Fall nimmt der Soundchip des Atari ST die Rückwandlung der digitalen Werte in Töne vor. Die Hardware des Samplers erübrigt sich dann für die Ausgabe.

Neben der Grundsoftware ist bereits ein weiteres Programm für den Sound Master erhältlich. Es nennt sich »Sample Graph Editor« und vereinfacht die Nachbearbeitung und Analyse eines aufgenommenen Klangs.

Das Programm stellt eine Aufnahme grafisch auf dem Bildschirm dar. Die so erhaltene Kurve gibt Aufschluß über die Charakteristik des Klangs. Das Programm ist mausorientiert und läuft über verschiedene Softwarefilter auf einfache Weise ab. Auch hier läßt sich die Aufnahme zur schnellen Kontrolle wieder über den internen Lautsprecher abhören. Für ein Arrangement neuer Klangvariationen dienen verschiedene Blockoperationen, die Teile verschieben, kopieren oder löschen.

Die Hauptanwendung des Sound Masters liegt in der akustischen Untermalung eigener Software. Titelmelodien oder gesprochene Kommentare für ein Programm machen immer Eindruck. Auch für Werbe- oder Demonstrationszwecke eignet sich der Sound Sampler gut. Laut Angaben des Herstellers ist bereits Software geplant, um die Klänge über ein Midi-fähiges Keyboard abzuspielen. Nicht zuletzt ist die Sprach- und Tonanalyse ein interessantes Gebiet.

Print Technik bietet den Sound Master für den ST für 598 Mark an. Eine Version für den Commodore Amiga ist in Planung. (Wolfgang Czerny/hb)

Der Happy-Digitizer: Prinzip der Digitalisierung

Begrüßt Ihr Computer Ihre Freundin mit einem freudigen »Hallo!«? Oder krächzt er manchmal wie Bruce Springsteen »Born in the USA«? Wenn nicht, dann fehlt ihm ein Sound-Digitizer zum Sprechen.

Ganz einfach ist es natürlich nicht, einem Computer den »guten Ton« beizubringen. Wie jeder weiß, kann er von Natur aus nur die beiden Zahlen Null und Eins verarbeiten, also »Strom« oder »nicht Strom«. Ganz im Gegensatz dazu stehen natürliche Töne, die auch alle Zwischengrößen zwischen zwei solchen Extremwerten kennen Solche fließenden Größen nennt man analog.

Wie kann man nun aber analoge Größen in computergerechte Signale umwandeln? Mit einem sogenannten Analog/Digital-Wandler.

Hierzu ein Vergleich: Der Arbeiter Udo hat in einem Wasserkraftwerk die Aufgabe, den Druck in der Wasserleitung zu messen. Da alle Druckmeßsysteme ausgefallen sind, stehen ihm nur ein Eimer und eine Stoppuhr zur Verfügung. Da der Eimer nur einen Meßstrich besitzt und in dem Wasserwerk alles sehr genau zugeht, darf er lediglich angeben, ob der Eimer leer oder voll ist.

Dies entspricht ganz der Situation des Computers.

Elektronik ist schnell

Der schlaue Udo dreht einen Wasserhahn auf und startet zugleich die Stoppuhr. Jetzt wird der Eimer so lange unter den Hahn gehalten, bis er voll ist, ausgeschüttet und ein Strich in einer Liste vermerkt. Dieser Vorgang wird so lange wiederholt, bis die Stoppuhr abgelaufen ist. Die Anzahl der Striche ist ein Maß für den Wasserdruck.

In der Elektronik ist es ganz genauso: Eine Spannung (Wasserdruck) steuert einen VCO (Voltage Controlled Oscillator = spannungsabhängiger Schwingkreis). Die abgegebenen Impulse während einer bestimmten Zeit (Stoppuhr) werden gezählt und sind eine direkte Angabe für die Spannung. Dieses Verfahren läßt sich noch verfeinern, indem man im übertragenen Sinne, anstatt den Eimer direkt zu füllen, erst einen Bottich eine bestimmte Zeit vollaufen läßt und aus diesem mit dem Eimer das Wasser abschöpft (Dual Slope). Der Vorteil dieses Verfahrens ist seine Einfachheit, die aber, wie man sieht, sehr viel Zeit in Anspruch nimmt. Ein Meßvorgang nach diesem Verfahren im Computerbereich dauert bis zu 0,1 sec.

Eine weitere simple Möglichkeit ist es, einen Meßzylinder eine bestimmte Zeit unter den Wasserhahn zu halten und die Füllhöhe abzulesen. Dies geht relativ schnell, allerdings benötigt man einen geeichten Meßzylinder. Einen mit Wasser gefüllten Zylinder abzulesen ist einfach, aber mit einem »elektronischen Zylinder« einen Computerchip zu lesen, stößt auf fast unüberwindliche Schwierigkeiten. So muß man für jeden einzelnen Teilstrich eine Vergleichseinheit (Komperator) einbauen. Strebt man zum Beispiel eine Auflösung von 1:1000 an, setzt das immerhin 1000 solcher Baugruppen voraus. Derart gebaute A/D-Wandler gehören somit zwar zu den schnellsten (Wandlerfrequenzen über 10 MHz), aber auch zu den teuersten (weit über 100 DM).

Für das dritte Verfahren benötigt der Arbeiter Udo eine Anzahl verschieden großer Eimer, von denen einer immer doppelt so groß ist wie der vorhergehende (1 Liter, 2 l, 4 l, 8 l, 16 l, ...). Das Wasser aus dem aufgefüllten Bottich wird nun in den größten Eimer gefüllt. War genügend Wasser vorhanden, ihn voll zu machen, kann man ihn zur Seite stellen, andernfalls kippt man das Wasser wieder zurück in den Sammelbehälter. Auf diese Weise verfährt man mit allen Eimern. Zum Schluß braucht man nur die Eimer der Reihe nach aufzustellen, und der Computerprofi sieht eine wohlbekannte Anordnung der Zahlen (128, 64, 32, 16, ...) vor sich, eine Binärzahl, wobei ein voller Eimer einer Eins und ein leerer einer Null entspricht. Auffällig ist, daß man bereits mit acht Eimern den Wasserdruck in 256 Stufen bestimmen kann, oder allgemein: mit N Eimern 2^n Stufen.

Das restliche Wasser, das entweder im Bottich übrigbleibt, oder das fehlt, den kleinsten Eimer zu füllen, nennt man Meßfehler.

Dieser kann durch Verwendung vieler Eimer vernachlässigbar klein gehalten werden. Da auch nur eine geringe Anzahl von Einzelversuchen durchgeführt werden muß, vereint diese Technik Schnelligkeit und Genauigkeit. Aus diesem Grund stellt sie auch das Nonplusultra bei Kleincomputern dar.

Damit nicht alles in grauer Theorie versinkt, wollen wir die Lehrstunde für Wasserwerksarbeiter abschließen, um uns den Beispielen für ein paar praktische Einsatzgebiete eines Digitizers zuzuwenden.

Messen mit dem Computer

Ein in der Praxis wichtiges Anwendungsgebiet ist die elektronische Meßwerterfassung. In jedem noch so kleinen Digitalmultimeter findet sich deshalb ein solcher A/D-Wandler.

Interessant für den Computerbesitzer wird es aber erst, wenn sich ein solches Gerät auch an seinen Computer anschließen läßt. Das ermöglicht es dem Computer, Vorgänge zu überwachen, nötigenfalls Alarm auszulösen oder Durchschnittsgrößen zu erfassen, wie zum Beispiel bei maschineller Produktion, Alarmanlagen und Wetteranalyse.

Wer genügend Geld investiert, um einen der ganz schnellen Wandler zu kaufen, kann damit sogar Videobilder einlesen und auswerten (Videodigitizer).

In diesem Sonderheft gehen wir jedoch auch auf ein anderes interessantes Anwendungsgebiet, die Sounddigitalisierung, näher ein.

Mit unserer kleinen Bauanleitung kann man sich Träume erfüllen, die aufgrund fehlender Hardware lange nicht machbar waren. Es steht jetzt eigenen Programmen mit Sprachausgabe oder Spielen mit digitalisierter Musik nichts mehr im Weg.

(R. Wagner/O. Strunk/Udo Reetz)



Der Happy-Digitizer: Anwendungsbeispiele

Fragt man sich nicht manchmal in der Disco, wie Paul Hardcastle sein »na na na na nineteen« stottert oder Jean Michele Jarre seine außerirdischen Klänge auf CD bannt? Die Lösung heißt »Sound-Digitizer«.

Dies ist nicht etwa ein neuer Song aus der italienischen Discoküche, sondern ein kleines unscheinbares Kästchen, das wahre Wunder vollbringt. Einige Musikbanausen mißbrauchen dieses Wunderkästchen auch dazu, Temperaturen zu messen, Klimadiagramme zu erzeugen oder Geschwindigkeiten zu bestimmen. Wir aber wenden uns den interessanten Anwendungsbereichen eines A/D-Wandlers, der Musik, zu. Das ganze Prinzip beruht im Grunde darauf, mit Hilfe dieses Kästchens Musik oder Klänge in einen Computer einzulesen, diese dann mit einem Programm zu verändern beziehungsweise zu verfälschen und schließlich wieder auszugeben. Eine der bekanntesten Methoden, das »Scratching«, welches früher wahre Artistenübungen an den Plattenspielern erforderte, kann jetzt jeder ausprobieren, ohne seinen Plattenspieler zu ruinieren. Ein Discjockey erklärte es uns folgendermaßen: »Nimmt man einen einzelnen Trommelschlag, hört man Poofff. Drehe ich nun die Platte rückwärts, erklingt fffooP. Durch schnelles Vor- und Rückwärtsdrehen des Plattentellers entsteht das typische Scratchgeräusch :fffooPoofffoo-Poo....«.

Versuchen Sie es einmal zu sprechen und dann mit dem zu vergleichen, was nachher mit unserem Digitizer-Programm herauskommt. Ein Computer wartet jedoch mit ein paar Effekten mehr auf. Einfach zu erklären, aber schwer zu realisieren ist die Tonhöhenmanipulation (Transponation), die jeder unabsichtlich schon einmal versucht hat, indem er eine Schallplatte mit der falschen Geschwindigkeit ablaufen ließ. Bei einem Computer geht dieser Vorgang ähnlich vor sich. Allerdings steht er vor einer fast unlösbaren Aufgabe, zum Beispiel bei der simultanen Transponation von Sprache. Das heißt, man spricht in ein Mikrophon, das am Computer angeschlossen ist, hinein, während der Lautsprecher das Gesagte in einer anderen Stimmlage wiedergibt. Das Dilemma des Computers besteht nun darin, die Sprachinformation schneller oder langsamer ablaufen zu lassen, ohne die Sprachgeschwindigkeit zu verändern. Das löst der Computer dadurch, daß er kurze Teile öfters hintereinander wiederholt oder kurze Abschitte überspringt (Elvis Presley mit Micky-Mouse-Stimme klingt aber doch recht seltsam). Nach einem wiederum einfachen Strickmuster sind Hall, Echo und Verzögerung realisiert. Der Computer dient als riesiges Schieberegister, er hat die Aufgabe, die Eingangsdaten verzögert zum Ausgang zu schieben. Mischt man das Ausgangssignal zum Teil wieder zum Eingangssignal, so entsteht der Echo- oder Halleffekt, abhängig von der Größe der Rückkopplung. Die Mischung zweier Digitalsounds geschieht durch einfache Mittelwertbildung (beide Soundkanäle werden addiert und das Ergebnis halbiert). Macht man bei der Programmierung der genannten Effekte Fehler, entstehen die allseits bekannten Roboter- und Geisterstimmen. Vereint man die bisher erwähnten Techniken miteinander, erhält man eine Sound-Sample-Maschine, wie Musiker sie verwenden: Eine Stimme oder ein Geräusch wird über ein Mikrophon digital aufgenommen und durch Transponation in alle Tonlagen überführt. Über ein angeschlossenes Keyboard kann man mit dem aufgenommenen Klang jede Melodie spielen oder Schlagzeugsolos (Digitaldrums) programmieren. Das Programm »Soundy« legt den Grundstock zu all diesen Klangzaubereien! Es besteht im wesentlichen aus zwei Teilen: dem Eingabeprogramm IN und dem Ausgabe- und Sampleprogramm OUT. Als dritten Teil könnte man noch die gemeinsam verwendeten Unterprogramme bezeichnen, welche die Aufgabe haben, die Sound-Ein- und -Ausgabe vorzubereiten (Interrupts, Initialisierung der Hardware).

Hier nun das Prinzip des Samplings: Es erfolgt keine kontinuierliche Ausgabe des ganzen Speichers, sondern sie wird anhand einer Befehlstabelle kontrolliert. Einzelne Teile werden in verschiedenen Geschwindigkeiten wiederholt ausgegeben. Ein besonderer Effekt, der in »Soundy« Anwendung findet, ist das Rückwärtsspielen, das ein Scratching ermöglicht. Dazu gibt man in der Tabelle fünf verschiedene Parameter an: Startadresse im Speicher, Endadresse (Longwords), Ablaufgeschwindigkeit (Word), (dabei entspricht 14 ungefähr der Originalgeschwindigkeit, kleinere Zahlen, bis 1, beschleunigen die Ausgabe und heben die Tonlagen an, und umgekehrt für tiefere Tonlagen), Anzahl der Wiederholungen (Word), wobei positive und negative Zahlen möglich sind. Zahlen kleiner Null lassen das Stück rückwärts abspielen. Es ist zu beachten, daß bei positiven Zahlen der Wert 0 einmaligem Abspielen entspricht, die 1 zweimaligem, etc. Bei negativen Zahlen beginnt einmaliges Abspielen bei -1 etc.

Jetzt fragt sich jeder Atari-Freak sicherlich, wie aus einem derart primitiven Soundgenerator überhaupt digitalisierter Sound herauszuholen ist. Nun, so einfach gestaltet sich das Ganze natürlich auch wieder nicht, darum reicht die Qualität auch nicht an die der professionellen Geräte heran. Aber immerhin: Der Sound-Chip ist zwar relativ einfach, doch besitzt er drei unabhängig steuerbare Lautstärkeregister, jedoch leider mit logarithmischem Aufbau. Das heißt, die Lautstärkewerte müssen mit Hilfe einer Tabelle delogarithmiert werden. Die Werte für ein Lautstärkeregister schwanken zwischen 0 und 15 und stellen damit eine 4-Bit-Auflösung dar. Durch Kopplung aller drei Register läßt sich diese auf 7 Bit steigern.

Das Programm soll nur den Grundstock zu weiteren eigenen Ideen legen, wobei die Ansteuerung der Hardware fast optimal gelöst wurde, da wir davon ausgehen, daß der normale Atari-Besitzer nicht die Erfahrung und Meßinstrumente (Oszi, Sinusgenerator) besitzt, die Tabelle oder andere spezielle Unterprogramme zu ändern. Außerdem ersetzt kein noch so gutes Programm die eigene Kreativität.

(R. Wagner/O. Strunk/Udo Reetz/hb)

Der Happy-Digitizer: Die Hardware

Einige technische Grundlagen zur Theorie der Sounddigitalisierung, eine Bauanleitung und eine Handvoll Bauteile verwandeln den Atari ST in ein Klangwunder. Mit dem Happy-Digitizer macht computern auf dem ST noch viel mehr Spaß!

Die Qualitäten eines A/D-Wandlers bestimmen genau zwei Größen: Das Auflösevermögen gibt an, in wie viele Stufen eine analoge Größe zerlegt werden kann und dient als Maß für die Genauigkeit des Wandlers. Da die digitale Größe eine Binärzahl darstellt, wird die Auflösung mit Hilfe der Bit-Anzahl der Digitalausgabe ausgedrückt. In der Musik gibt dieses Bit-Maß indirekt auch die zu übertragende Dynamik wieder. Weil leise Stellen mit Hilfe von geringen und laute mit großen Amplituden übertragen werden, muß ein Wandler die lauten wie die leisen Stellen in möglichst viele Stufen unterteilen, um die Verzerrungen so gering wie möglich zu halten. Dabei sollte das Auflösevermögen um so feiner sein, je größer das Verhältnis zwischen der lautesten und der leisesten Musikstelle ist. So erfordert beispielsweise die Übertragung von klassischer Musik ein wesentlich besseres Wandlersystem, als Popmusik. Am besten eignen sich vokale Musikstücke zur Digitalisierung auf einem Computer. Die Übertragungsdynamik eines A/D-Wandlers errechnet sich aus der Anzahl der Bits mal 6.

Einige Beispiele:

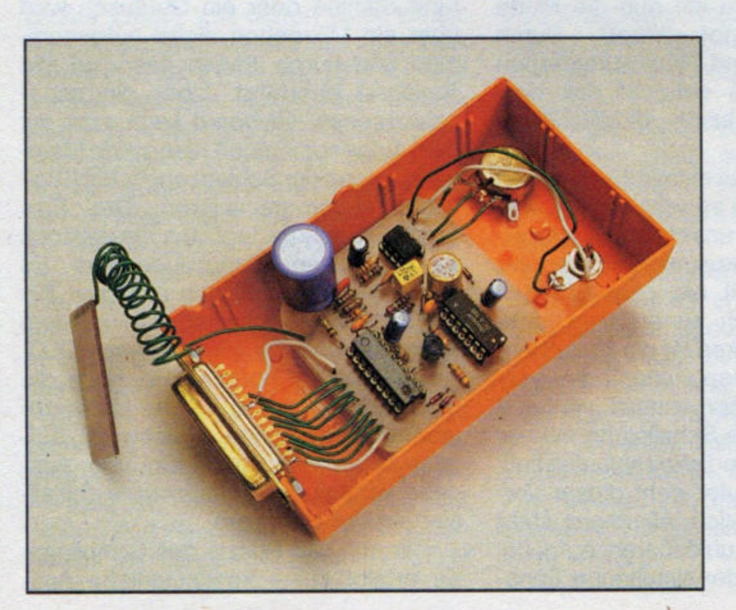
- Ein gutes Kassettenlaufwerk mit einer qualitativ hochwertigen Kassette überträgt eine Dynamik von bis zu 60 db (Dezibel).
- Ein normaler Plattenspieler schafft bis zu 80 db.
- Ein modernes Hallgerät in einer Gesangsanlage löst Töne mit 12 Bit auf und besitzt somit einen Dynamikumfang von 72 db.
- Das Nonplusultra stellen die CD-Plattenspieler mit mehr als 100 db Dynamik dar. Übrigens, der Digitizer arbeitet ähnlich einem solchen Gerät.

Weiterhin gilt: je besser das Auflösevermögen und damit der Dynamikumfang, desto hochwertiger die Übertragungsqualität; je geringer das Grundrauschen, desto geringer der Klirrfaktor. Dieser errechnet sich so:

$$\text{Klirrfaktor} = \frac{1}{\text{Anzahl der Quantisierungsstufen}} = 2^{-\text{Bit-Anzahl}}$$

Als zweites Qualitätsmerkmal eines Digitizers dient seine Wandlerfrequenz. Dabei gilt der Grundsatz, daß die Wandlerfrequenz mindestens doppelt so groß wie die höchste zu übertragende Frequenz sein muß. Dies leuchtet sehr leicht ein, wenn man sich deutlich macht, daß sich eine Schwingung aus einem positiven und einem negativen Schwingungsabschnitt zusammensetzt. Um nun diese Schwingung übertragen zu können, muß der Wandler bei jeder Halbwelle mindestens eine Abtastung vornehmen, das heißt, er muß zwei Abtastungen pro Vollwelle durchführen (doppelte Frequenz). Was passiert aber, wenn man versucht, mit einer geringen Wandlerrate zu hohe Frequenzen zu übertragen? Es entstehen dann unkontrollierte Schwingungen, sogenannte Spiegelfrequenzen. Spiegelfrequenzen deshalb, weil diese Schwingungen in einem Frequenzspektrum liegen, das an der halben Wandlerfrequenz gespiegelt wurde. Liegt zum Beispiel eine Umsetzfrequenz von 20 kHz vor und man versucht, einen Ton mit 9 kHz einzulesen, entsteht ein weiterer Ton mit 11 kHz. Dieser Effekt tritt sowohl bei der Analog/Digital- als auch bei der Digital/Analog-Wandlung auf. Deshalb sind einerseits zwei Tiefpaßfilter notwendig, um hohe Töne »abzuschneiden«.

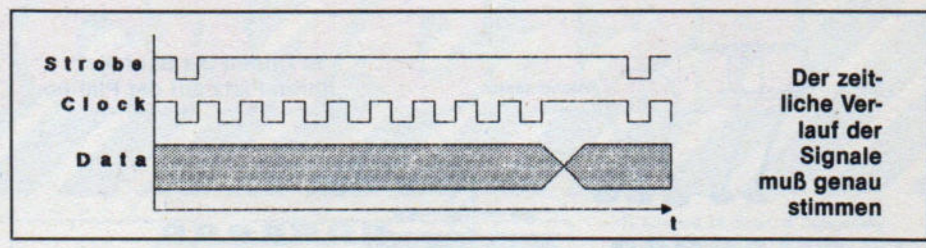
Das menschliche Ohr vermag andererseits Töne bis zu einer Frequenz von 20000 Hz wahrzunehmen. Dies bedeutet, daß ein A/D-Wandler, der das ganze Spektrum übertragen soll, mit einer Abtastrate von über 40 kHz zu arbeiten hat. Moderne Digitallaufwerke besitzen sogar eine Abtastrate von 43 kHz. Aufgrund längerer Untersuchungen stellte man fest, daß ein Zuhörer zwischen einer mit 32 kHz und einer mit 43 kHz aufgelösten Musik nicht unterscheiden kann. Bei Sprachausgabe reicht sogar eine Umsetzfrequenz von 8 kHz vollkommen aus. Da bei einer digitalen Aufnahme jeder anfallende Wert gespeichert werden muß – zum Beispiel sind bei einer Abtastfrequenz von 43 kHz, 16-Bit-System und Stereo 192 KByte (!) pro Sekunde nötig –, wird klar, warum die Frequenz auf das absolute Minimum reduziert wird. Der hier vorgestellte Wandler besitzt eine 8-Bit-Auflösung und eine maximale Frequenz von 30 kHz.



Der fertige Digitizer findet in einem kleinen Kunststoffgehäuse Platz. Links ist deutlich die kleine Platine erkennbar, die in den Modulschacht gesteckt wird.



In den Modulschacht gesteckt, übernimmt diese kleine Platine die Stromversorgung



Der zeitliche Verlauf der Signale muß genau stimmen

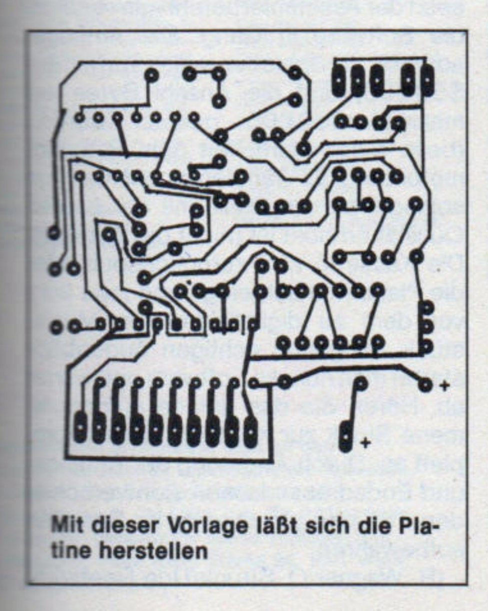
Dies scheint auf den ersten Blick ein bißchen wenig, aber man beabsichtigt ja nicht, irgendwelche Musikstücke in Hi-Fi-Qualität abzuspeichern, sondern diese zu verzerren und zu verändern. Deshalb kommt es auch nicht darauf an, sie möglichst naturgetreu wiederzugeben, sondern möglichst viel zu speichern. Als ein guter Kompromiß erwies sich beim Atari ST eine 8-Bit-Auflösung und eine Abtastfrequenz von 20 kHz. Eine höhere Umsetzfrequenz ergibt auch deshalb keinen Sinn, weil der Monitorlautsprecher hohe Töne nicht übertragen kann. Dies hat zugleich den Vorteil, daß er für die gefürchteten Spiegelfrequenzen als Tiefpaßfilter wirkt. Eine noch niedrigere Umsetzrate macht diese unangenehmen Pfeiftöne schon hörbar. Reserviert man 680 KByte als Datenspeicher, reicht das, etwa 35 Sekunden Musik aufzunehmen.

Nun zur Hardware im einzelnen. Der Kern unseres A/D-Wandlers ist der ZN 427 von Ferranti, ein etwa 30 Mark teures IC, das ohne aufwendige Zusatzbeschaltung auskommt und problemlos erhältlich ist. Hier die Beschreibung der einzelnen Anschlüsse:

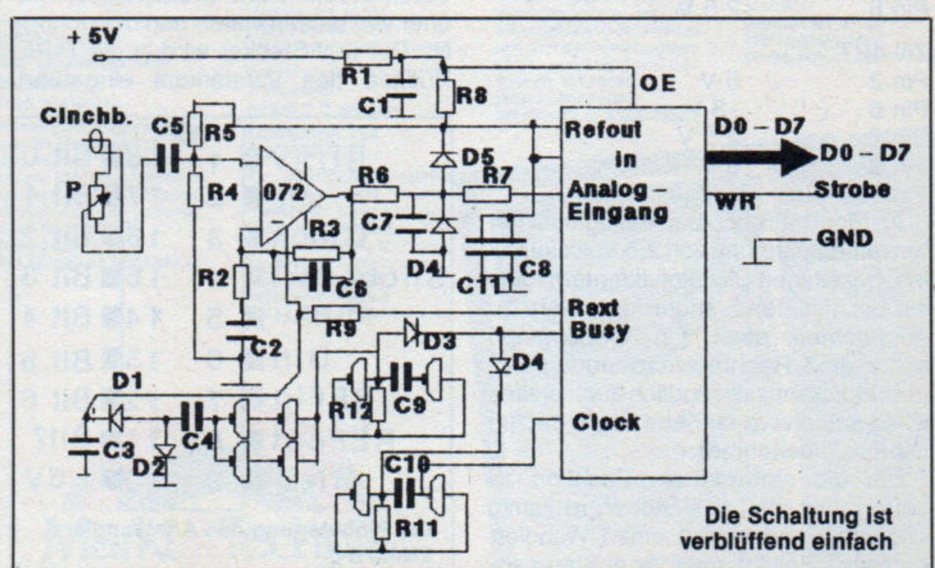
Pin 1, Busy (Low aktiv): Während des Umsetzvorganges wird Busy auf Low gesetzt, um dem Computer mitzuteilen, daß die Daten ungültig sind.
Pin 2, OE (Output Enable): Bei Low gehen die Datenausgänge in den hochohmigen Zustand über (bei SOUNDY nicht gebraucht).
Pin 3, Clock-Eingang: Steuert Wandlungsvorgang im IC, es werden pro Wert 9 Taktimpulse benötigt (8+1).
Pin 4, Wr: Bei Low Impuls wird die Wandlung gestartet.
Pin 5: Negative Versorgungsspannung.
Pin 6: Analogeingang (0 V=$00 bis 2,56V=$FF).
Pin 7, Refin: Referenzspannungseingang, er bestimmt den Wandlerbereich (hier: mit Refout verbunden).
Pin 8, Refout: Referenzspannungsausgang 2,56 Volt.
Pin 9: Masse Eingang.
Pin 10: Spannungsversorgung.
Pin 11 bis 18 Digitalausgänge (Tristate).

Für die einwandfreie Funktion unterstützen den Wandler drei Hilfsschaltungen. Ein Vorverstärker (TL 072) verstärkt das Eingangssignal bis es für das Wandler-IC brauchbar ist.

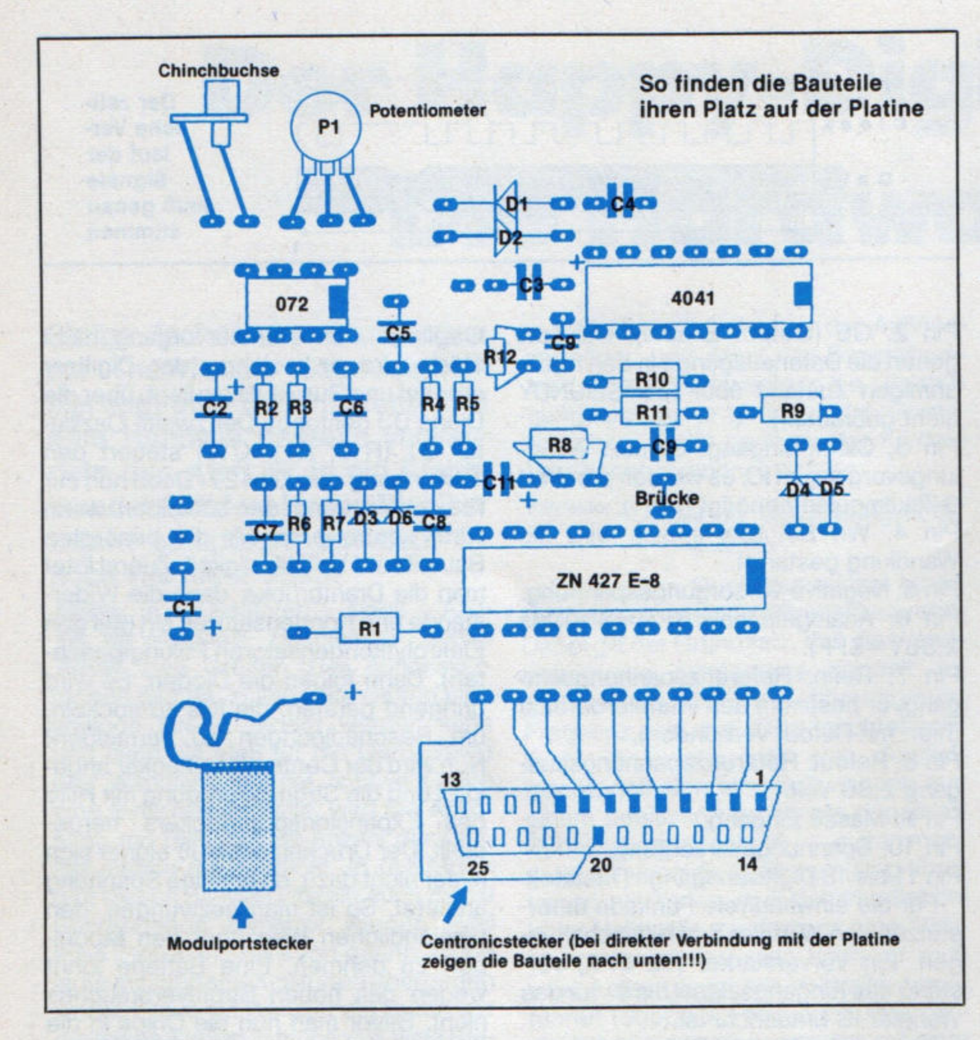
Dieser ist mit einer Nullpunktkorrektur (R4, R5) und einem Tiefpaßfilter (R2, R3, C6) versehen. Der Oszillator O2 (R11, C9) hat die Aufgabe, die negative Versorgungsspannung zu liefern. Er erzeugt hierzu erst einmal eine Wechselspannung (C4), die gleichgerichtet (D1, D2) eine negative Vorspannung von -3 bis -4 Volt liefert. Damit der Oszillator den Wandlervorgang nicht stört, wird er, solange der Digitizer arbeitet und Busy auf Low liegt, über die Diode D3 gestoppt. Der zweite Oszillator O1 (R10, R11, C10) steuert den Takteingang des ZN 427. Doch nun zur Praxis. Während der Lötkolben warm wird, kontrollieren wir die besorgten Bauteile auf Vollständigkeit. Zuerst lötet man die Drahtbrücke, dann die Widerstände und Kondensatoren ein (bei den Elekrolytkondensatoren Polung beachten). Dann folgen die Dioden. Es wird dringend geraten, die ICs zu sockeln, um Beschädigungen zu vermeiden! Nun wird der Centronics-Stecker angelötet und die Stromversorgung mit Hilfe des Expansionsportsteckers hergestellt. Der Druckeranschluß eignet sich leider nicht dazu, da er keine Spannung anbietet. So ist man gezwungen, den umständlichen Weg über den Modulport zu nehmen. Eine Batterie lohnt wegen des hohen Stromverbrauches nicht. Bevor man nun die Chips in die Fassungen steckt, wird der Stecker am Computer und der Modulportstecker mit dem Computer verbunden. Jetzt den Computer einschalten. Funktioniert er nicht einwandfrei, sofort ausschalten und die Platinen überprüfen, besonders, ob der Modulportstecker keine anderen Pins als den links oben berührt. Die Pins der einzelnen ICs

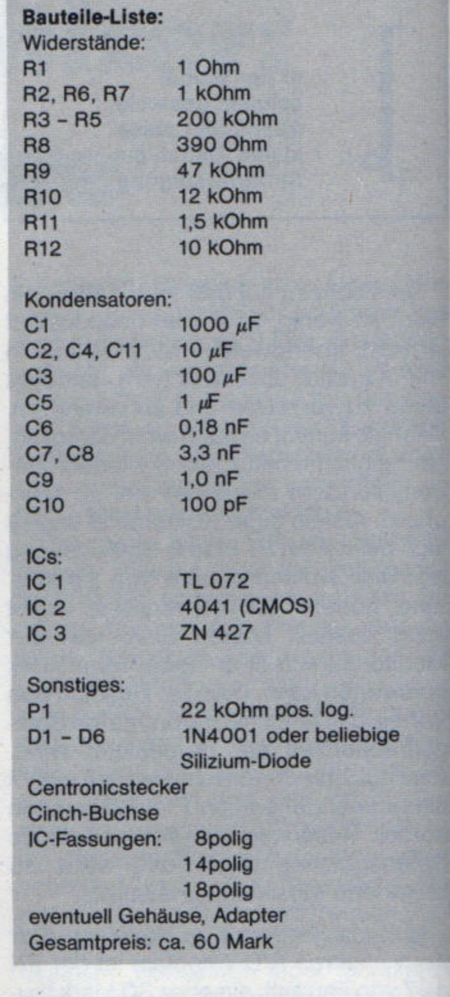


Mit dieser Vorlage läßt sich die Platine herstellen



Die Schaltung ist verblüffend einfach





Bauteile-Liste:

Widerstände:

R1	1 Ohm
R2, R6, R7	1 kOhm
R3 – R5	200 kOhm
R8	390 Ohm
R9	47 kOhm
R10	12 kOhm
R11	1,5 kOhm
R12	10 kOhm

Kondensatoren:

C1	1000 μF
C2, C4, C11	10 μF
C3	100 μF
C5	1 μF
C6	0,18 nF
C7, C8	3,3 nF
C9	1,0 nF
C10	100 pF

ICs:

IC 1	TL 072
IC 2	4041 (CMOS)
IC 3	ZN 427

Sonstiges:

P1	22 kOhm pos. log.
D1 – D6	1N4001 oder beliebige Silizium-Diode

Centronicstecker
Cinch-Buchse
IC-Fassungen: 8polig, 14polig, 18polig
eventuell Gehäuse, Adapter
Gesamtpreis: ca. 60 Mark

müssen folgende Spannungen aufweisen (Masse des Meßgerätes mit Minus-Anschluß verbinden):

IC 4041:

Pin 14	5 V
alle anderen	0 V

IL 072:

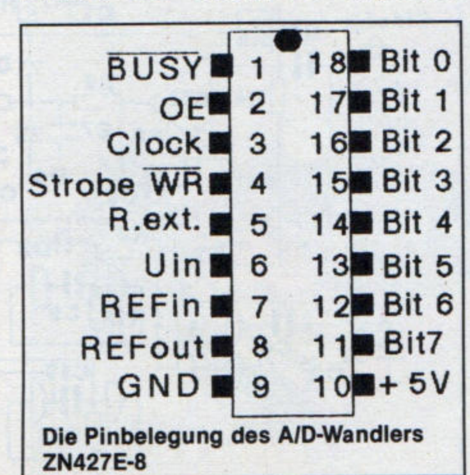
Pin 4	–3 V bis –4 V
Pin 8	5 V
Pin 5	2,5 V

ZN 427:

Pin 2	5 V
Pin 5	–3 V
Pin 6	2,5 V
Pin 9	0 V
Pin 10	5 V

An Pin 3 muß sich bei Analogmultimetern eine Spannung von 2,5 Volt einstellen, bei einigen Digitalmultimetern kann ein sich laufend ändernder Wert zu beobachten sein. Ein Oszillograph sollte eine Rechteckschwingung zeigen. Nun kann man endlich auch dieses IC an seine vorgesehene Stelle setzen (Kerbung beachten).

Um die einwandfreie Funktion zu testen, lädt man das Steuerprogramm »Soundy« und startet einen Wandlervorgang. Besitzt man eine Stereoanlage mit Cinch-Anschlüssen, braucht man nur den A/D-Wandler mit der TAPE-OUT-Buchse des Verstärkers zu verbinden. Dabei spielt es keine Rolle, ob man den rechten oder linken Kanal verwendet. Komplizierter wird es, wenn nur DIN-Ausgänge zur Verfügung stehen. Man besorgt sich deshalb einen Adapter-DIN-Stecker von DIN- auf vier Cinch-Stecker, und probiert aus, welcher von diesen vieren nun der richtige ist. Der DIN-Stecker wird in die TAPE-Buchse des Verstärkers eingeführt.



Die Pinbelegung des A/D-Wandlers ZN427E-8

Dabei erweist es sich als sehr vorteilhaft, wenn sich an den Verstärker zwei Tape-Decks anschließen lassen. Andernfalls verbindet man den A/D-Wandler mit dem TAPE-Anschluß, und den Kassettenrecorder legt man an den Auxiliary-Eingang. Nachdem man das Musikstück, das man digitalisieren möchte, an der lautesten Stelle startet, setzt der Assemblerbefehl »gin« endlich die Software in Gang. Die Anfangsadresse im Speicher sollte vorher auf $50000, und die Anzahl Bytes auf maximal 680000 gesetzt werden (beim Mega-Atari). Mit dem Potentiometer des A/D-Wandlers sucht man nun solange, bis die Stelle mit der besten Qualität erreicht ist (nicht die lauteste). Die Kassette wird zurückgespult oder die Platte neu aufgelegt, und zwar kurz vor dem zu digitalisierenden Musikstück. Im genau richtigen Augenblick startet man nun die Software und wartet ab. Hören Sie das gerade aufgenommene Stück zur Kontrolle einmal komplett an. Durch Änderung der Anfangs- und Endadresse lassen sich verschiedene Stücke gleichzeitig im Speicher aufbewahren.

(R. Wagner/O. Strunk/Udo Reetz/hb)

Der Happy-Digitizer Steuersoftware

Endlich geschafft! Ist der A/D-Wandler zusammengebaut, muß man nur noch das Steuerprogramm »Soundy« eingeben. Diese Anleitung hilft dabei, Fehler zu vermeiden.

Das Source-Listing ist im Seka-Format abgedruckt. Seka ist ein weit verbreiteter Assembler. Er eignet sich für unsere Zwecke sehr gut, da man Änderungen im Programm sofort ausprobieren kann, ohne umständlich zwischen Assembler, Editor und Linker hin- und herwechseln zu müssen.

Um das Listing einzugeben, löscht man vorsichtshalber den Source-Speicher mit dem »ks«-Befehl (kill source). »t1« wählt die erste Zeile an und »i« schaltet in den Insertmodus (beenden mit »esc«); dann kann die Tipperei losgehen. Die Kommentare (alles hinter einem Strichpunkt) brauchen Sie natürlich nicht einzugeben. Sie dienen lediglich zum besseren Verständnis des Programms. Vor allem bei der Eingabe der Datentabellen sollte man besondere Sorgfalt walten lassen, da hier bei einer fehlerhaften Zahl der Seka keine Fehlermeldung ausgibt! Ist die Eingabe abgeschlossen, assembliert man probehalber das Programm (»a« und Return; bei der Frage nach »Options« Return drücken). Erscheint die Meldung »No Errors«, ist alles in Ordnung. Wenn nicht, rufen Sie mit »t« und Zeilennummer die entsprechende Zeile auf und editieren sie mit »e«. »w« speichert das fehlerfreie Programm dann auf Diskette.

Wie schon erwähnt, besteht Soundy eigentlich aus zwei Programmen, dem Eingabe- und dem Ausgabeteil. Da zu Beginn der Speicher noch leer ist, nimmt man zuerst etwas auf.

Dies geht folgendermaßen vor sich: Den fertigen Digitizer in den Centronics-Port (=Druckerport) des Atari stecken, und das Kabel der Stromversorgung mit Pin 1 (+5 Volt) links oben im Modulschacht (linke Seite des Computers) verbinden. Da die Hardware des Atari sehr empfindlich reagiert, geben Sie acht, daß keine anderen Pins mit dem Kabel in Berührung kommen. Hat man einen Stecker für den Schacht gebastelt, braucht man natürlich keine Angst zu haben, sofern man die richtigen Pins verbunden hat. Vor dem Umstecken der Verbindungen empfiehlt es sich, den Computer auszuschalten. Der Centronics-Stecker des Digitizers kann ohne Bedenken aus- und eingesteckt werden, auch wenn der Computer in Betrieb ist.

Jetzt verbindet man nur noch den Digitizer mit der Tonquelle (Stereoanlage, Radio, Kassettenrecorder, etc.), was natürlich das passende Kabel mit den richtigen Adaptern voraussetzt.

Vorsicht beim Eintippen!

Tip: Den Kopfhörerausgang benutzen, da dort noch die Klangfarbe verändert werden kann (zum Beispiel mehr Bässe oder weniger Höhen, damit ein eventuell vorhandenes Rauschen besser unterdrückt wird). Auch das Tonsignal wird noch über die Endstufe vorverstärkt. Damit ist die Hardware bereit!

Nun schaltet man den Computer an, bootet das Betriebssystem (möglichst wenig Accessories, da später noch viel freier Speicherplatz benötigt wird), lädt »Seka« und anschließend den gespeicherten Sourcecode mit dem »r«-Befehl. Als nächstes gibt man im Quellprogramm die Startadresse eines freien Speicherbereiches für die Aufnahme der Sounddaten an. Gewöhnlich wählt man $50000 (im Listing bereits eingetragen). Dort beginnt bei normalem (RAM-)TOS und den üblichen zwei Accessories der Systemdiskette der freie Speicherplatz. Da der Assembler, sowie das Quellprogramm und der Objektcode vor dieser Adresse gespeichert sind, steht nun der ganze Speicher bis zum Ende des RAMs zur freien Verfügung, abzüglich der letzten 32 KByte. Dieser Bereich ist für den Bildschirmspeicher des Atari ST reserviert. Die Obergrenze liegt somit beim Mega-Atari bei $F7FFF und beim Atari ST 260 bei $77FFF. Somit bleiben beim ST 260 nur $77FFF-$50000=$27FFF (zirka 158000 dezimal), und beim »großen« Atari $F7FFF-$50000=$A7FFF (zirka 683000 dezimal) Byte frei. Als Besitzer des ROM-TOS kann man die Startadresse natürlich niedriger wählen. Am besten »erkundigt« sich jeder selbst nach dem Assemblierungsvorgang beim Seka-Assembler mit »h«, an welcher Stelle der Objektcode endet. Zu dieser Adresse wird noch eine Sicherheitszone von $2000 addiert und das Ergebnis als neue Startadresse verwendet. Nachdem noch die Anzahl der zu digitalisierenden Daten zwei Zeilen tiefer angegeben wurde, assembliert man das Programm (»a«). Nun beginnt endlich die Eingabe:

Mit »gin« (hier ist nicht das Getränk gemeint, sondern ein Programmstart beim Label »in«) und »bp« als Breakpoint (bei 2. Breakpoint Return drücken) startet das Einleseprogramm. Der Cursor steht jetzt still oder ist gar nicht zu sehen. Die Tastatur gibt keinen Klick mehr von sich, und aus dem Monitorlautsprecher (nicht aus der Stereoanlage) erklingt das Lied, das Sie gerade abspielen. Hören Sie aber nichts, so kann das mehrere Gründe haben:

- Das Radio, die Stereoanlage etc. ist nicht eingeschaltet, oder das entsprechende Ausgabegerät wurde nicht selektiert (falls man das Tonsignal am Kopfhörerausgang abnimmt, ist dies leicht festzustellen),
- der Volume-Regler an Ihrem Monitor ist zu leise eingestellt (zum Testen bitte voll aufdrehen),

Wenn der ST »sprachlos« bleibt

- die Steckverbindungen (Tonquelle-Digitizer-Computer) haben einen Wackelkontakt (leichtes Rauschen oder Brummen),
- die Stromversorgung des A/D-Wandlers mit Pin 1 des Modulschachtes ist nicht richtig angeschlossen (insbesondere dann, wenn man gar nichts hört).

Scheiden alle diese Fehlerquellen aus, so liegt entweder ein Fehler in der

Hardware vor (Chip defekt, etc.), oder das Programm wurde falsch abgetippt.

Ertönt jedoch nur ein Rauschen im Lautsprecher, das vielleicht noch entfernt an das Originallied erinnert, so ist das ganz normal: Mit dem Potentiometer am Digitizer stellt man den optimalen Punkt ein (dort, wo sich der Sound am besten anhört), und regelt dann, sofern man den Digitizer an den Kopfhörerausgang angeschlossen hat, noch am Verstärker Höhen, Tiefen, sowie die Lautstärke nach. Hierbei gilt: Lieber ein bißchen über- als untersteuert.

Ist das Ergebnis immer noch schlecht und gelingt es nicht, trotz mehrfachem Nachregeln das Rauschen zu unterdrücken (die Musik sollte aber schon gut kommen!), so liegt entweder immer noch ein Fehler in der Hardware/Software vor, oder das Lied läßt sich sehr schwer digitalisieren (zum Beispiel Klassik, Schlagzeugsolos), oder der Monitorstecker ist nicht richtig eingesteckt. Das Grundrauschen des Fernsehlautsprechers kann durch Software natürlich nicht einfach weggezaubert werden.

Der hier vorgestellte Digitizer digitalisiert volle acht Bit. Wenn man einen D/A-Wandler (das Gegenstück zu einem A/D-Wandler, er wandelt Zahlen in analoge Werte um) am Atari St anschließt, so kommt man in den Genuß der vollen acht Bit.

Acht Bit machen Musik

Es besteht prinzipiell die Möglichkeit, mehr als nur ein Stück an verschiedene Speicherpositionen zu legen, was besonders beim Mischen und Erzeugen eigener Klangwerke zum Tragen kommt. Wurde nun ein Stück digitalisiert, so kann man den Speicherbereich vom Seka aus mit »wi« als Image speichern. Hierbei verlangt Seka die Anfangsadresse der Sounddaten (in unserem Beispiel $50000) sowie die Endadresse ($50000+Anzahl der Bytes). Später kann der Anwender jederzeit die Daten mit »ri« und der Angabe der Start- und Endadresse in den Speicher laden. Bitte beachten Sie, daß auf einer zweiseitigen formatierten Diskette »nur« 720 KByte Platz haben, was bei einer maximalen Anzahl von 680000 Datenbyte (Mega-Atari mit RAM-TOS) nicht viel Platz für ein weiteres Musikstück übrig läßt.

Sind die ersten Testläufe bestanden und alle Regler bestens eingestellt, so kann man nach Herzenslust digitalisieren. Hierzu ein Tip beim Aufnehmen: den Anfang des Liedteiles, das man digitalisieren möchte, mehrfach anhören, wieder an den Anfang zurückspulen, in den Computer »gin« sowie »bp« eintippen, damit man nur noch die Return-Taste drücken muß, um das Programm zu starten. Jetzt betätigt man den Abspielknopf des Kassettenrecorders oder senkt den Tonarm des Plattenspielers ab und wartet, den Finger über der Return-Taste, bis der inzwischen gut eingeprägte Anfang kommt, und drückt, – am besten kurz zuvor, damit auch ja nichts verlorengeht – die Return-Taste.

Wie kann man sich nun das Ganze anhören? Dazu muß man etwas über die







Funktionsweise des zweiten Programmteiles wissen. Wie schon erwähnt, gibt Soundy nicht nur Eingelesenes einfach wieder, sondern es kann auch »samplen«. Das heißt, mit verschiedenen Geschwindigkeiten beliebig oft rückwärts oder vorwärts unterschiedlich lange Musikstücke abspielen. Zu Beginn testen wir jedoch nur, wie sich der digitalisierte Sound anhört. Dazu tragen Sie am Ende des Programmes in der »sampletab« (=Sample-Tabelle, siehe Listing) einige Werte ein.

Zuerst kommt der Assemblerbefehl »dc.l« (=declare data, longwords), der dem Assembler signalisiert, daß jetzt ein oder mehrere Langwörter (das sind Adreßzeiger in den Speicher) folgen. In unserem Fall sind das die Start- und die Endadresse der Sounddaten. Wurde $50000 als Startadresse und als Anzahl 100000 Byte gewählt, so gibt man in der ersten Zeile nach dem Label »sampletab« die Anweisung »dc.l $50000,$50000+100000« ein.

Sound samplen

Wie zu sehen ist, wurde die Endadresse nicht ausgerechnet, sondern die Eigenschaft des Assemblers benutzt, einfache Rechnungen selber auszuführen. Soundy muß jetzt aber noch wissen, in welcher Geschwindigkeit die Ausgabe erfolgen soll, und wie oft es den, durch obige Adressen festgelegten Teil, wiederholen soll. Deshalb benötigt es noch eine zweite Zeile mit Daten. Die Normalgeschwindigkeit hat den Wert 14. Wir wollen unser Werk einmal hören. So wird als Wiederholungszahl eine Null eingegeben. Die zweite Anweisungszeile sieht demnach folgendermaßen aus: »dc.w 14,0«.

Die einzelnen Werte müssen unbedingt durch Kommas getrennt werden und alle vier Werte dürfen nicht gleichzeitig (!) in eine Zeile geschrieben werden. Am Ende der Sample-Tabelle steht immer folgende Zeile: »dc.l -1,-1«.

Dies ist das Zeichen, daß die Tabelle beendet und somit das Programm abzuschließen ist. Vergißt man diese Zeile, so spielt das Programm immer weiter, bis es im Speicher auf die Zahlen -1,-1 stößt. Unter Umständen ist das nie der Fall, und der Computer »erwacht« nur noch durch einen Druck auf den Resetknopf und erneutes Laden zum Leben.

Nach der Eingabe dieser drei Zeilen (da sie im abgedruckten Listing bereits enthalten sind, reicht es, die Adressen und Werte entsprechend anzupassen)

```
;=====================================================================
;=                          S O U N D Y                              =
;=          6-BIT SOUND-DIGITIZER für ATARI ST COMPUTER              =
;=          -------------------------------------------              =
;=                              by                                   =
;=                                                                   =
;=      Rolf Wagner             &          Oliver Strunk             =
;=      Sulzbrunn 3                        Am Göhlenbach 57          =
;=      8961 Sulzberg                      8960 Kempten              =
;=                                                                   =
;=      (Software)                         (Hardware)                =
;=                                                                   =
;=====================================================================
;
;
;-----------------------------------
; sound-chip-register initialisieren
;-----------------------------------
reg_init:
move.l #sound,a0
sound_loop:
 move.w (a0)+,d0
 bmi reg_init_exit
 move.b d0,register
 move.w (a0)+,d0
 move.b d0,value

 jsr output
 bra sound_loop

reg_init_exit:
 rts
;
;-----------------------------
; sound-chip-register ausgabe
;-----------------------------
output:
 move.b register,d1
 cmp.b #7,d1
 bne sound_out2
 move.b d1,$ffff8800
 move.b $ffff8800,d2
 and.b #%11000000,d2
 or.b value,d2
 move.b d2,$ffff8802
 rts

sound_out2:

 move.b register,$ffff8800
 move.b value,$ffff8802
 rts

;-----------------------------------------------
; programmende: interrupts etc wiederherstellen
;-----------------------------------------------
bpp:
 move.b save7,$fffffa07
 move.b save9,$fffffa09
 move.w #$2308,sr
 move.l stackptr,-(sp)
 move.w #$20,-(sp)
 trap #1
 add.l #6,sp

 move.w #5,d0
wait_2:
 move.w #$ffff,d1
wait_1: dbra d1,wait_1
 dbra d0,wait_2
bp:
 rts


;-----------
; variablen
;-----------
register: blk.w 1,0
value: blk.w 1,0


 stackptr: blk.l 1,0

 mem_start:blk.l 1,0
 mem_end:blk.l 1,0
```

```
save7: blk.w 1,0
save9: blk.w 1,0

sound_sign: blk.w 1,0
speed: blk.w 1,0

;-----------------------------------------
; initialisierungs-werte für sound-chip
;-----------------------------------------
sound: dc.w  0,255, 1,255, 2,255, 3,255, 4,255, 5,255, 6,0, 7,%00111111
       dc.w  8,0, 9,0, 10,0, -1,-1


;------------------------------------------
; lautstärke-stufen für lautstärke-register
;------------------------------------------
poketab:
 dc.l $08000000 ,$09000000, $0a000000      ; 00
 dc.l $08000000 ,$09000000, $0a000200      ; 01
 dc.l $08000000 ,$09000000, $0a000300      ; 02
 dc.l $08000200 ,$09000200, $0a000200      ; 03
 dc.l $08000500 ,$09000000, $0a000000      ; 04
 dc.l $08000500 ,$09000200, $0a000000      ; 05
 dc.l $08000600 ,$09000100, $0a000000      ; 06
 dc.l $08000600 ,$09000200, $0a000100      ; 07
 dc.l $08000700 ,$09000100, $0a000000      ; 08
 dc.l $08000700 ,$09000200, $0a000000      ; 09
 dc.l $08000700 ,$09000300, $0a000100      ; 0a
 dc.l $08000800 ,$09000000, $0a000000      ; 0b
 dc.l $08000800 ,$09000200, $0a000000      ; 0c
 dc.l $08000800 ,$09000300, $0a000100      ; 0d
 dc.l $08000800 ,$09000400, $0a000100      ; 0e
 dc.l $08000900 ,$09000000, $0a000000      ; 0f
 dc.l $08000900 ,$09000200, $0a000000      ; 10
 dc.l $08000900 ,$09000300, $0a000100      ; 11
 dc.l $08000900 ,$09000400, $0a000100      ; 12
 dc.l $08000900 ,$09000500, $0a000000      ; 13
 dc.l $08000900 ,$09000500, $0a000200      ; 14
 dc.l $08000900 ,$09000600, $0a000000      ; 15
 dc.l $08000900 ,$09000600, $0a000200      ; 16
 dc.l $08000a00 ,$09000200, $0a000000      ; 17
 dc.l $08000a00 ,$09000200, $0a000200      ; 18
 dc.l $08000a00 ,$09000400, $0a000100      ; 19
 dc.l $08000a00 ,$09000500, $0a000000      ; 1a
 dc.l $08000a00 ,$09000500, $0a000200      ; 1b
 dc.l $08000a00 ,$09000600, $0a000100      ; 1c
 dc.l $08000a00 ,$09000600, $0a000300      ; 1d
 dc.l $08000b00 ,$09000100, $0a000000      ; 1e
 dc.l $08000b00 ,$09000200, $0a000100      ; 1f
 dc.l $08000b00 ,$09000300, $0a000100      ; 20
 dc.l $08000b00 ,$09000400, $0a000100      ; 21
 dc.l $08000b00 ,$09000500, $0a000100      ; 22
 dc.l $08000b00 ,$09000600, $0a000000      ; 23
 dc.l $08000b00 ,$09000600, $0a000200      ; 24
 dc.l $08000b00 ,$09000700, $0a000000      ; 25
 dc.l $08000b00 ,$09000700, $0a000100      ; 26
 dc.l $08000b00 ,$09000700, $0a000300      ; 27
 dc.l $08000b00 ,$09000700, $0a000400      ; 28
 dc.l $08000b00 ,$09000800, $0a000100      ; 29
 dc.l $08000b00 ,$09000800, $0a000300      ; 2a
 dc.l $08000b00 ,$09000800, $0a000400      ; 2b
 dc.l $08000b00 ,$09000800, $0a000500      ; 2c
 dc.l $08000b00 ,$09000800, $0a000500      ; 2d
 dc.l $08000c00 ,$09000200, $0a000000      ; 2e
 dc.l $08000c00 ,$09000200, $0a000200      ; 2f
 dc.l $08000c00 ,$09000400, $0a000100      ; 30
 dc.l $08000c00 ,$09000500, $0a000000      ; 31
 dc.l $08000c00 ,$09000500, $0a000300      ; 32
 dc.l $08000c00 ,$09000600, $0a000000      ; 33
 dc.l $08000c00 ,$09000600, $0a000200      ; 34
 dc.l $08000c00 ,$09000700, $0a000000      ; 35
 dc.l $08000c00 ,$09000700, $0a000300      ; 36
 dc.l $08000c00 ,$09000700, $0a000400      ; 37
 dc.l $08000c00 ,$09000800, $0a000000      ; 38
 dc.l $08000c00 ,$09000800, $0a000300      ; 39
 dc.l $08000c00 ,$09000800, $0a000400      ; 3a
 dc.l $08000c00 ,$09000800, $0a000500      ; 3b
 dc.l $08000c00 ,$09000900, $0a000000      ; 3c
 dc.l $08000c00 ,$09000900, $0a000300      ; 3d
 dc.l $08000c00 ,$09000900, $0a000400      ; 3e
 dc.l $08000c00 ,$09000900, $0a000500      ; 3f
 dc.l $08000c00 ,$09000900, $0a000500      ; 40
```

Listing »Soundy«, der Digitizer für den Atari ST

wird das Programm neu assembliert. (nicht vergessen: Nach jeder noch so kleinen Änderung muß das Programm assembliert werden, da man sonst beim Starten immer noch den alten Objektcode im Speicher hat!)

Endlich kann man nun mit »gout« (entspricht »go out«, also »starte Ausgabeprogramm«) und der gewohnten Eingabe von »bp« als Breakpoint das digitalisierte Stück anhören. Um die Fähigkeiten von Soundy richtig zu begreifen, dringen wir nun schrittweise in die Geheimnisse der »sampletab« vor:

Erhöhen wir die Wiederholungszahl von Null auf Eins. Neu durchassemblieren und starten: Wie zu erwarten, ertönt das Stück zweimal. Nun suchen wir durch Ändern der Start- und Endadressen eine Stelle in dem Musikstück, die zum Beispiel einen Trommelschlag wiedergibt. Gehen Sie beim Ändern der Adressen erst in 20000-Byte-Schritten (Grobeinstellung) und dann in 3000-Byte-Schritten vor (Feineinstellung). Später kann man dann mit noch kleineren Intervallen arbeiten. Zu Beginn ist das jedoch wenig sinnvoll, da bereits ein Sprung von 1000 Byte kaum mehr wahrnehmbar ist. Haben wir eine geeignete Stelle gefunden, so spielen wir nur 10000 Byte, diese jedoch 10mal.

Also könnten die Sampledaten folgendermaßen aussehen:

```
dc.l
$50000+43000,$50000+43000+10000
dc.w 14,10
```

Wie man sieht, kann man auch mehrere Zahlen addieren. Das vereinfacht das Ändern der Adressen in dem ohnehin unkomfortablen Editor des Assemblers ein wenig.

Trommelwirbel

Nach dem Start ertönt nun ein maschinengewehrähnlicher Sound, vorausgesetzt, es wurde ein Trommelschlag als Teilstück gewählt. Nur mit Adressen und einer Wiederholungszahl läßt sich schon eine ganze Menge unterschiedlicher Soundeffekte erzeugen. Beispielsweise eine Kombination aus der ersten und der zweiten Liedversion. Eine Sampletabelle dazu könnte folgendermaßen aussehen (um die alten Zeilen ganz zu löschen, wird der Befehl »z« und die Zeilenanzahl verwendet):

```
dc.l $50000,$50000+300000
(ganzes Lied, 300 kb lang)
dc.w 14,0
```



SPITZEN-SOFTWARE FÜR ATARI ST

WordStar/MailMerge

Version 3.0 mit MailMerge

Der Bestseller unter den Textverarbeitungsprogrammen bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.
WordStar/MailMerge ist an den ATARI ST bereits fertig angepaßt und läßt sich bequem über Funktionstasten steuern.
WordStar/MailMerge für den ATARI ST wird auf zwei 3½-Zoll-Disketten geliefert. Sie beinhalten: CP/M-Z80-Emulator, WordStar/MailMerge-Dateien.
Hardwareanforderungen: ATARI-ST-Computer, 80-Zeichen-Monitor, ein 3½-Zoll-Diskettenlaufwerk, beliebiger Drucker mit Centronics-Schnittstelle.



(sFr. 178,–/öS 1890,–*)

Bestell.-Nr. MS 106

Dazu die richtige Literatur: WordStar für ATARI ST

Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit WordStar ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.

Best.-Nr. MT 90208
ISBN 3-89090-208-1
DM 49,– (sFr. 45,10/öS 382,20)

dBASE II

Version 2.41

Das bedienerfreundliche Datenbanksystem dBASE II beinhaltet eine eigene Programmiersprache für die Erstellung von individuellen Bildschirmmasken. dBASE II wurde unter dem Betriebssystem GEM/TOS für den ATARI ST angepaßt und unterstützt die Schnelligkeit des 68000-Prozessors.
dBASE II läßt sich komfortabel über Pull-down-Menüs mit der Maus steuern.
dBASE II für den ATARIST wird auf einer 3½-Zoll-Diskette geliefert.
Hardware-Anforderungen: ATARI-ST-Computer, 80-Zeichen-Monitor, 3½-Zoll-Diskettenlaufwerk.

DM 348,–* (sFr. 295,–/öS 2980,–*)

Bestell.-Nr. MS 306

* inkl. MwST. Unverbindliche Preisempfehlung

Dazu die richtige Literatur: dBASE II für ATARI ST

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch von dem deutschen Erfolgsautor Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen, die nur von einem Profi stammen können, bei Ihrer täglichen Arbeit mit dBASE II. Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten.

Best.-Nr. MT 90206
ISBN 3-89090-206-5
DM 49,– (sFr. 45,10/öS 382,20)

Markt & Technik-Softwareprodukte erhalten Sie in den Fachabteilungen der Kaufhäuser, in Computershops oder im Buchhandel.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Wenn Sie direkt beim Verlag bestellen wollen:
Gegen Vorauskasse durch Verrechnungsscheck oder mit der abgedruckten Zahlkarte.
Bestellungen im Ausland bitte an untenstehende Adressen:
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. 042/41 56 56
Österreich: Ueberreuter Media Verlagsges. mbH, Alser Straße 24, A-1091 Wien, Tel. 0222/48 15 38-0

```
;-------------------------------------------------------------
; hier beginnt das hauptprogramm: einsprung nach 'in' bzw 'out'
;-------------------------------------------------------------
;
;
;------------------- digi-sound-input ----------------------------
;
;
in:
 clr.l -(sp)                    ; supervisor-modus einschalten
 move.w #$20,-(sp)              ;        "
 trap #1                        ;        "
 add.l #6,sp                    ;        "
 move.l d0,stackptr             ;        "

 jsr reg_init                   ; sound-chip register initialieren

 move.l #$50000,d0              ; speicher-startadresse (=$50000)
 move.l d0,mem_start
 add.l  #500000,d0              ; anzahl bytes zu samplen (500 kb)
 move.l d0,mem_end              ; (max bis speicherobergrenze)

 move.b $fffffa07,save7         ; interrupts ausschalten
 move.b $fffffa09,save9         ;        "
 move.b #0,$fffffa07            ;        "

 move.b #%01000000,$fffffa09    ;        "
 and.w #%1111100011111111,sr    ;        "
 or.w #$500,sr                  ;        "

 clr.l d2                       ; ersten strobe - high senden
 move.b #$20,d2                 ; damit wird der digitizer
 move.b #14,$ffff8800           ; gestartet
 move.b $ffff8800,d3            ;
 or.b d2,d3                     ;
 move.b d3,$ffff8802            ;

 move.b #7,$ffff8800            ; port b des sound-chips
 move.b $ffff8800,d0            ; für die eingabe vorbereiten
 and.b #%01111111,d0            ;        "
 move.b d0,$ffff8802            ;        "

 move.l mem_start,a2            ; speicher-start
 move.l #poketab,a1             ; startadresse der lautstärke-werte


;-------- eigentliche eingabe-hauptschleife

inloop:
 mulu d6,d6
 mulu d6,d6                     ; ein bißchen warten ....

 clr.l d5
 move.b #15,$ffff8800           ; register 15 des sound-chips selektieren
 move.b $ffff8800,d5            ; und sound-daten holen (reg 15 = port b)

 move.b #14,$ffff8800           ; strobe low ausgeben
 move.b $ffff8800,d1            ;        "
 and.b #$df,d1                  ;        "
 move.b d1,$ffff8802            ;        "
 move.b $ffff8800,d1            ; strobe high ausgeben
 or.b #$20,d1                   ;        "
 move.b d1,$ffff8802            ;        "

 move.b d5,(a2)+                ; daten in speicher schreiben
 cmp.l mem_end,a2               ; speicher-ende ??
 bgt bpp                        ; ja, zum programm-ende

                                ; sound gleich ausgeben...
 and.w #%11111100,d5            ; nur bits 2-7
 move.w d5,d1                   ; multiplikation mal 3 (2+1)
 lsl.w #1,d1                    ;        "
 add.w d1,d5                    ;        "
 movem.l (a1,d5),d1-d3          ; register-tripel laden
 movem.l d1-d3,$ff8800          ; daten in sound-register schreiben
                                ; damit man mit-hören kann...

 bra inloop                     ; zurück zum anfang
;---------------- digi-sound-output ----------------------------
out:
 clr.l -(sp)                    ; supervisor-mode einschalten
 move.w #$20,-(sp)              ;        "
 trap #1                        ;        "
```

Listing »Soundy« (Fortsetzung)









```
dc.l
$50000+43000,$50000+43000+10000
( nur Trommelschlag)
dc.w 14,10
dc.l  $50000,$50000+300000
(nochmal ganzes Lied)
dc.w 14,0
...
```

Um das Ganze noch etwas exotischer zu gestalten, ist bei Soundy ein besonderer Programmpunkt eingebaut: das Rückwärtsspielen.

Das erfordert lediglich ein Minuszeichen vor dem Wert für die Zahl, wie oft ein Stück zu wiederholen ist. Einziger Unterschied: Beim Rückwärtsspielen wird die tatsächliche Anzahl angegeben, also -1 für einmal rückwärts, -2 für zweimal und so weiter.

Rückwärts für den heißen Rhythmus

Durch den »Rückwärtssound« eröffnen sich noch mehr Bereiche des Samplings.

Damit stehen dem Anwender Tür und Tor offen für die Entwicklung neuer Klänge. Ein Beispiel dafür ist das »Scratchen«.

Doch für denjenigen, dem das alles nicht genug ist, hält Soundy noch ein Schmankerl bereit. Die Geschwindigkeit kann (fast) beliebig gewählt werden. Man braucht nur den Geschwindigkeitswert in der Sample-Tabelle verändern. Erhöht man diesen Wert, so verlangsamt sich die Soundausgabe, der Ton wird tiefer. Reduziert man ihn (immer größer Null!), so beschleunigt sich die Ausgabe. Dieser Effekt macht es möglich, ganze Stücke, insbesondere Schlagzeugbegleitung, zusammenzustellen. Unser Beispiel aus der Sample-Tabelle:

```
dc.l $50000,$50000+300000
dc.w 14,0 (Normalgeschwindigkeit)
dc.l $50000,$50000+300000
dc.w 20,0  (langsam)
dc.l $50000,$50000+300000
dc.w 3,0 (superschnell)
....
```

Es lassen sich Effekte, wie das Anlaufen eines Plattenspielers oder Jaulen eines schlechten Kassettenrecorders, spielend realisieren. Mischt man nun alle machbaren Effekte, entstehen wahre Meisterwerke.

Alle diese Informationen versetzen Sie in die Lage, Ihre eigenen Klangkompositionen zu verwirklichen. Die Redaktion wartet gespannt auf das Resultat...

(R. Wagner/O. Strunk/Udo Reetz/hb)

```
 add.l #6,sp                     ;              "
 move.l d0,stackptr              ;              "

 jsr reg_init                    ; sound-chip register initialisieren

 move.b $fffffa07,save7          ; interrupts ausschalten
 move.b $fffffa09,save9          ;              "
 move.b #0,$fffffa07             ;              "
 move.b #%01000000,$fffffa09     ;              "
 and.w #%1111100011111111,sr     ;              "
 or.w #$500,sr                   ;              "

;------ hier beginnt die eigentliche sample-hauptschleife --------

 move.l #poketab,a1              ; adresse der lautstärkewerte
 move.l #sampletab,a0            ; adresse der sample-daten

sample_repeat:
 move.l (a0)+,mem_start          ; sample-start-adresse
 bmi bpp                         ; bei -1 abbruch
 clr.w sound_sign                ; sound-sign = 1 für rückwärts-sound
 move.l (a0)+,mem_end            ; sample-end-adresse
 move.w (a0)+,speed              ; ausgabe-geschwindigkeit
 move.w (a0)+,d7                 ; anzahl der wiederhohlungen
 bpl repeat                      ; bei >0 normal spielen
 move.w d7,sound_sign            ; sound-sign setzten
 move.w #-1,d5                   ; vorzeichen ändern (wiederholungsanzahl)
 sub.w d7,d5                     ;              "
 move.w d5,d7                    ; wiederholungsanzahl nach d7

repeat:
 move.l mem_start,a2             ; daten-start-adresse
 move.l mem_end,a3               ; daten-end-adresse

outloop:
 move.w speed,d6                     ; warteschleife
out_wait: dbra d6,out_wait       ;      "

 clr.l d5
 tst.w sound_sign                ; test auf rückwärts-sound
 bne back_sound                  ; ja !!

 move.b (a2)+,d5                 ; daten aus speicher lesen
 cmp.l a3,a2                     ; speicher-ende ???
 bgt end_repeat                  ; ja !!

sound_cont:
 and.w #%11111100,d5             ; nur bits 2-7
 move.w d5,d1                    ; multiplikation mal 3 (2+1)
 lsl.w #1,d1                     ;              "
 add.w d1,d5                     ;
 movem.l (a1,d5),d1-d3           ; register-tripel laden
 movem.l d1-d3,$ff8800           ; daten in sound-register schreiben

 bra outloop                     ; zurück zum anfang

back_sound:
 move.b -(a3),d5                 ; rückwärts-sound
 cmp.l a2,a3                     ;              "
 blt end_repeat                  ;              "
 bra sound_cont                  ;              "

end_repeat:
 dbra d7,repeat                  ; sound wiederholen
 bra sample_repeat               ;              "

;-------------------------------------------------------------------
;      sample-data : startdresse (long), endadresse (long)
;                    geschwindigkeit (word)(normal 14)
;                    anzahl der wiederholungen (word)
;                    ( positiv: 0 = einmal spielen
;                               1=  zweimal   "    , usw.
;
;                      negativ:-1 = einmal rückwärts spielen
;                              -2 = zweimal    "        "    , usw.)
;-------------------------------------------------------------------
sampletab:
dc.l $50000,$50000+500000       ; startadresse,endadresse
dc.w  14,0                      ; geschwindigkeit, wiederholungsanzahl

dc.l -1,-1                      ; ende der daten-liste (nicht vergessen!!)

end
```

Listing »Soundy« (Schluß)

Helfer für die Schreibtischtäter

Wirklich perfekte Textverarbeitung gibt es auf dem Atari ST genausowenig wie auf anderen Computern. Vier Programme bieten sich an, die Zensuren des ST in Schönschrift, Rechtschreiben und Schreibgeschwindigkeit in Zukunft zu verbessern. Wir konnten insgesamt gute Noten vergeben.

Tatort München, Redaktionsgebäude eines Verlages, Freitagnacht 23.37 Uhr mitteleuropäische Sommerzeit! Alles wirkt einsam und verlassen. Nur aus einem Fenster in der vierten Etage (es ist das vierte von rechts) dringt flimmernder Lichtschein in die finstere Nacht. Plötzlich zerreißt ein gellender Schrei die idyllische Stille. Splitterndes Glas, ein dunkler Schatten durchbricht das Fenster und stürzt in die Tiefe, gefolgt von Bahnen mattweißen Endlospapiers mit Mikroperforation. Raschelnd falten sich Berge von Computerausdrucken über einen reglosen männlichen Körper: Redakteur Karl-Friederich Stumpfental hat endgültig die Schlacht mit dem Textcomputer verloren. In der geballten Faust hält er einen Fetzen Computerpapier. Darauf sind die seltsamen Worte zu lesen: '..ur mignstige Krfte knnen w..'. Was wollte uns Karl-Friederich Stumpfental mit dieser geheimnisvollen letzten Botschaft mitteilen? Für erfahrene ST-Benutzer sind die letzten Worte unseres wackeren Wortklaubers schnell entschlüsselt. Karl-Friederich Stumpfental war Mitglied der »ÖÄÜß-Gesellschaft zur Foerderung der perfekten deutschen Textverarbeitung auf dem Atari ST e.V.«. Wie die kriminalpolizeiliche Untersuchung ergab, hatte ihn bei der Erprobung eines neuen revolutionären Textverarbeitungsprogrammes ein Schlaganfall ereilt, weil dieses Prunkstück deutscher Programmiererkreativität zum Drucken die Texthardcopy-Routinen des TOS benutzt hatte...

Unsere kleine Geschichte ist selbstverständlich frei erfunden, weder Karl-Friederich Stumpfental noch die ÖÄÜß-Gesellschaft haben je existiert. Sehr wohl aber existieren Textverarbeitungsprogramme in Hülle und Fülle, darunter gar manche, die mit der deutschen Sprache ihre Probleme haben. Auch der Atari ST, durch seinen hervorragenden Monochrom-Monitor SM124 für die Arbeit mit Texten geradezu prädestiniert, hat auf diesem Gebiet immer noch seine Schwierigkeiten. Trotz einiger schmerzlich vermißter Funktionen hat sich für komfortable Textverarbeitung auf ST-Computern inzwischen das Programm »1st Word« von GST aus England durchgesetzt. Aufgrund seiner weiten Verbreitung ist dieses sehr bedienungsfreundliche Programm praktisch zum Textverarbeitungsstandard bei den ST-Benutzern geworden. So ist es wohl auch zu erklären, daß drei unserer vier Probanden das Markenzeichen »1st« in ihrem Namen tragen. Bei »1st Spooler« handelt es sich um eine Softwarelösung für einen Druckerspooler mit besonderen Eigenschaften. »1st Mailmaster« bietet eine Serienbrief-Schnittstelle zwischen »1st Word« und »DB-Master One«. Als besonderer Leckerbissen ist »1st Lektor« hervorzuheben. Sein Programmierer wäre zu Recht beleidigt, wenn dieses Programm als simpler Speller (Programm zur Rechtschreibkorrektur) bezeichnet würde. Das vierte Programm in unserer Sammlung mit Namen »LQ-Font« verspricht Schönschriftdruck in verschiedenen Schrifttypen mit einigen weitverbreiteten Matrixdruckern.

In der Folge seien die vier Programmpakete genauer im einzelnen vorgestellt:

Zum Lieferumfang von »LQ-Font« gehört neben der Programmdiskette ein zehnseitiges Beiheft, das in knapper, informativer Form Aufgabe und Funktion des Programmes erklärt. Die Diskette enthält vier LQ-Font-Programme, jeweils eines für Epson-Drucker, für Drucker mit IBM-Modus, für den Atari SMM804 und für die Star-Drucker Gemini und Delta. Neben den Dateien mit den Zeichengeneratoren für fünf verschiedene Schrifttypen findet man noch eine Desktop-Accessory »PRINTER.ACC« zur bequemen Steuerung von LQ-Font.

Dies ist ein kleiner Beispieltext für
das Programm LQ-FONT.

©®™ÅæÆ ∫∫∫ ĳĲאבגדהוזחטיכלמנסעפצקרש
αβΓπΣσµ∫∫∫ τΦΘΩδ∮φ∈√$^{n\,2\,3}$

Ende des Textes!!!!

Dies ist ein kleiner Beispieltext für
das Programm LQ-FONT.

©®™ÅæÆ ∫∫∫ ĳĲאבגדהוזחטיכלמנסעפצקרש
αβΓπΣσµ∫∫∫ τΦΘΩδ∮φ∈√$^{n\,2\,3}$

Ende des Textes!!!!

Dies ist ein kleiner Beispieltext für
das Programm LQ-FONT.

©®™ÅæÆ ∫∫∫ ĳĲאבגדהוזחטיכלמנסעפצקרש
αβΓπΣσµ∫∫∫ τΦΘΩδ∮φ∈√$^{n\,2\,3}$

Ende des Textes!!!!

Dies ist ein kleiner Beispieltext für
das Programm LQ-FONT.

©®™ÅæÆ ∫∫∫ ĳĲאבגדהוזחטיכלמנסעפצקרש
αβΓπΣσµ∫∫∫ τΦΘΩδ∮φ∈√$^{n\,2\,3}$

Ende des Textes!!!!

LQ-Font – schöner schreiben auch ohne NLQ-Drucker

Die Funktionsweise von LQ-Font basiert auf ähnlichen Prinzipien wie die Bildschirmausgabe des Atari ST. Der ST-Bildschirm ist eine reine Grafikausgabeeinheit. Auch Text wird pixelweise grafisch abgebildet. LQ-Font behandelt den angeschlossenen Drucker in ähnlicher Weise. Im Computer wird ein grafischer Zeichengenerator aufgebaut, der von den verschiedensten Programmen oder bei der Texthardcopy aus dem GEM-Desktop durch ASCII-Codes angesteuert wird. Von all dem merkt der Drucker aber noch gar nichts.

LQ-Font setzt nämlich diese ASCII-Codes um und beliefert den Drucker ausschließlich mit Codes zur Steuerung der einzelnen Nadeln seines Druckkopfes. Der Drucker arbeitet also im Grafikbetrieb.

Schöne Schrift mit Standardnadeln

Daher ist es auch möglich, unabhängig vom Zeichengenerator des angeschlossenen Druckers mit LQ-Font bis zu 10 verschiedene Schrifttypen normal, fett, kursiv, breit und unterstrichen darzustellen. Auch Kombinationen dieser Darstellungsarten lassen sich einstellen. Der Schönschriftmodus kann im Desk-Menü des GEM-Desktop (Funktion »Drucker-Anpassung«) ein- oder abgeschaltet werden. Bei abgeschalteter Schönschrift sorgt LQ-Font für die Umsetzung der Atari-spezifischen Zeichencodes in die entsprechenden Codes für den angeschlossenen Drucker.

Die Qualität der Schönschrift ist sehr ansprechend und kann mit der sogenannten NLQ-Schrift (Near-Letter-Quality) der meisten Neun-Nadel-Drucker durchaus mithalten.

Dies gilt auch in bezug auf die Druckgeschwindigkeit. Die Grafik-Hardcopy-Funktion des TOS arbeitet einwandfrei. LQ-Font harmonierte mit fast allen getesteten Programmen. Bisher gab es nur Probleme mit der Vorabversion des neuen 1st Word plus und mit 1st Spooler. Zu einem Preis von 149 Mark bekommt man ein Programm, das ungeahnte Qualitäten aus einem normalen Matrixdrucker herausholt.

Drucker gehören bekanntlich zu den lahmen Schnecken unter den Peripheriegeräten. Wenn längere Texte ausgedruckt oder Bildschirm-Hardcopies angefertigt werden, so geht oft nichts mehr. Der Drucker werkelt vor sich hin, und der Computer dreht Däumchen. Kaffeepause ist angesagt! Besonders lang geraten solche Zwangspausen, wenn ein Typenrad-Drucker oder gar eine umgebaute Schreibmaschine mit Text versorgt werden will.

Doch einen so speicherstarken Computer wie den Atari ST ficht selbst ein solches Ärgernis nicht an. Der ST legt seine Daten in einem reservierten Speicherbereich ab, füttert den Drucker mit dem ersten Datenpaket und wendet sich wieder wichtigeren Aufgaben zu. Ab und an schaut er nach, ob sein langsamer Kollege seine Portion wohl richtig verdaut hat. Trifft dies zu, so schiebt er ihm das nächste Bündel in den Schlund. Ein guter Computer wie der ST erledigt diese Nebenaufgabe so schnell, daß der Computerbenutzer davon fast gar nichts bemerkt.

Ein derartiges Programm nennt man Softwarespooler. 1st Spooler ist beileibe nicht der erste ST-Spooler auf dem Markt, verfügt jedoch über einige Eigenschaften, die ihn aus der Masse seiner Konkurrenten hervorheben. 1st Spooler besteht aus den zwei Teilprogrammen »SPOOLER.ACC« und »INITSP.TTP« sowie einer Datei »TRTSTD« zur Umwandlung von Atari-Zeichencodes in ASCII-Codes für den Drucker. Die Umwandlungstabelle ist in einer Dialogbox leicht modifizierbar. Modifizierte Tabellen kann man speichern und laden. Auf Wunsch bleibt die Codewandlung auch bei direktem Druck ohne Spooler aktiv.

Drucken ohne Wartezeit

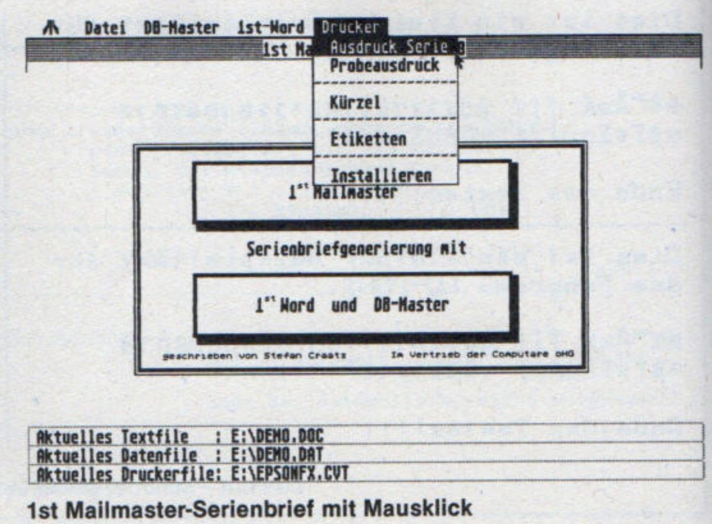
»SPOOLER.ACC« wird wie alle GEM-Desktop-Accessories bei Systemstart initialisiert, muß sich also beim Einschalten oder Reset des Computers auf der Startdiskette oder auf Partition C einer Harddisk befinden. Der Spooler ist jetzt zwar aktiviert, nimmt aber noch keinen Pufferspeicher in Anspruch. Der Puffer muß erst noch mit »INITSP.TTP« reserviert werden. Die Puffergröße ist frei wählbar. Nach der Pufferfestlegung kann 1st Spooler aus jedem GEM-Programm aufgerufen und benutzt werden.

1st Spooler ist GEM-unterstützt und wird über eine komfortable Dialogbox gesteuert. Die auszudruckende ASCII-Datei läßt sich in ein Dialogfeld eintragen oder aus einer Auswahlbox durch Anklicken festlegen. Man kann jeweils nur eine Datei zum Ausdruck vormerken. Die Ausgabe erfolgt je nach Einstellung der Druckeranpassung im Desk-Menü über den Parallel- oder den RS-232-Port des ST. Der Druckvorgang läßt sich nach erneutem Klick auf den Eintrag »SPOOLER« im Desk-Menü unterbrechen oder abbrechen.

Eine bei Softwarespoolern auf dem ST bisher einzigartige Fähigkeit stellt das Spoolen von Bildschirm-Hardcopies



1st Spooler-Hardcopies ohne Wartezeit



1st Mailmaster-Serienbrief mit Mausklick

dar. Nach Anklicken des Feldes »HCS« in der Bedienungsbox wird beim Drücken von »ALTERNATE/HELP« die Ausgabe der Grafikdaten an den Drucker über den Spoolerpuffer abgewickelt. Diese Funktion ist leider nur möglich, wenn sich das Betriebssystem im RAM befindet.

Alle besprochenen Funktionen arbeiten einwandfrei, Fehlbedienungen wie etwa Anklicken von »HCS« mit Betriebssystem im ROM werden zuverlässig abgefangen und erzeugen aussagekräftige Warnboxen. 1st Spooler ist ein sehr komfortabler Softwarespooler mit besonderen Eigenschaften wie Hardcopyspooling und der Ausgabe über die serielle Schnittstelle. Ihr ST wird sich für dieses Geschenk durch Arbeitseifer ohne Druckerpause bedanken.

Schreibfaule Computerbenutzer werden es kaum glauben können. Da soll es doch Zeitgenossen geben, die nicht nur gelegentlich mal einen netten Brief an die Erbtante Else schicken, um Genaueres über die angegriffene Gesundheit der guten Tante zu erfahren. Nein, es soll Geschäftsleute geben, die Hunderte von gleichlautenden Briefen an ebensoviele verschiedene Menschen schicken und diesen Personen durch strategisch günstig im Text verteilte Einstreuungen von persönlichen Anreden wie »...so möchte ich Sie, sehr geehrter Herr Sowieso, ganz besonders herzlich...« oder »...so möchte ich Sie, sehr verehrte Frau Weißnichtwie, ganz besonders herzlich...« vorgaukeln, daß dieser Brief ein wirkliches Einzelstück ist und nur für Herrn Sowieso oder Frau Weißnichtwie individuell entworfen wurde.

Diese fromme Lüge in der Kundenbetreuung hat den harmlos klingenden Namen Serienbrief. Serienbriefe gehören zu den Erfindungen im Geschäftsleben, die vor Einführung von computerunterstützten Textsystemen die bemitleidenswerten Sekretäre und Sekretärinnen zu menschlichen Schreibautomaten degradierten, weil ja damals wirklich jeder Brief eigenhändig geschrieben werden mußte. Mit einem guten Computer, einem Textverarbeitungsprogramm und einer gut organisierten Datenbank läßt sich diese Arbeit heutzutage ohne allzu großen Aufwand erledigen. Der Serientext wird zunächst ein einziges Mal im Textprogramm geschrieben und an den Stellen, an denen die persönlichen Textelemente erscheinen sollen, geheimnisvolle Steuerzeichen eingefügt. Aus der großen Datenbank sucht man sich Adresse, Anrede und Name des Ansprechpartners beim Kunden heraus und faßt diese persönlichen Textelemente in einer sogenannten Report-Datei zusammen.

Jedem dieser Elemente wird eines der Steuerzeichen zugeordnet. Ein sogenanntes Serienbriefprogramm sorgt nun dafür, daß der Brief immer wieder aus der Textdatei ausgedruckt wird und an den Stellen mit den Steuerzeichen die entsprechenden Elemente der einzelnen Datensätze aus der Report-Datei eingefügt werden.

Damit sind ohne geisttötende Menschenarbeit eine große Zahl von persönlich aussehenden Briefen entstanden. Nur falten und in die Briefumschläge stecken muß man sie noch von Hand, da das Falt- und Einkuvertierungsprogramm erst noch programmiert werden muß.

Überbrückungshilfe für Serientäter

Zweifellos ist der Atari ST ein guter Computer, der mit 1st Word über eine blitzsaubere Textverarbeitung verfügt. DB-Master one gehört auch nicht unbedingt in die Reihe der unbrauchbaren Datenbanken in der ST-Software. Zum ganzen Glück des Serienbriefschreibers fehlt also nur noch ein Serienbriefprogramm, das die Kluft zwischen den beiden Programmen wirkungsvoll überbrückt.

Die Lösung unseres Problems heißt 1st Mailmaster und kommt aus Berlin. Zum Lieferumfang gehören ein kurzes Bedienungshandbuch, eine Programm-Diskette und ein kleiner Stecker für den Joystickport 1 rechts hinten am ST. Dieses mechanisch wenig stabile Steckerchen bildet den Kopierschutz des Programmes. 1st Mailmaster ist normal kopierbar und läuft gleichermaßen auf Harddisk und Floppy-Disk. Allerdings muß der Stecker im Joystickport 1 stecken.

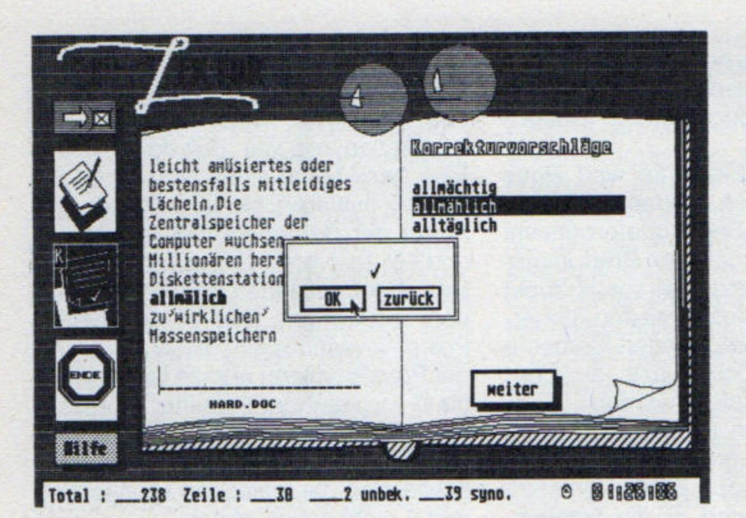
Dort bleibt er am besten, denn aufgrund seiner wirklich großartigen Konstruktion bekommt man ihn nur nach längerer Fummelei mit Schraubendreher und Präzisionszange wieder heraus. Auf älteren Exemplaren der ST-Computer soll es gelegentlich trotz Stecker Programmabstürze gegeben haben. Besitzer solcher Geräte sollten die Funktionsfähigkeit vor Kauf des Programmes erst einmal überprüfen. Auf der Programmdiskette befinden sich neben vier Demo-Dateien die zwei Dateien des eigentlichen Programmes und ein Druckertreiber für Epson-Drucker. Nach Doppelklick auf »MAILMAST.PRG« erscheint auf dem Monitor das typische Bild einer GEM-Applikation mit vier Pull-down-Menüs (das Desk-Menü nicht mitgezählt). Die beiden mittleren Menüs dienen dem Aufruf der Datenbank DB-Master one und der Textverarbeitung 1st Word. Die Laufwerke, auf denen diese Programme gesucht werden, sind in den beiden Pull-down-Menüs einstellbar. Leider hat der Programmierer nur die Laufwerke A bis D vorgesehen. Bei einer professionellen ST-Anlage mit zwei Diskettenlaufwerken und einer Harddisk mit vier Partitions sind die logischen Laufwerke E und F nicht ansprechbar.

Das Menü »Datei« bietet Funktionen zur Verwaltung, Konvertierung und Auswahl der für das Programm notwendigen Dateien. Die Reportdateien aus DB-Master one müssen vor Benutzung im Mailmaster mit der Funktion »Konvertierung« umgewandelt werden. Der Druckertreiber wird aus einer ».HEX«-Datei für 1st Word mit der Menüfunktion »Installieren« im Pull-down-Menü »Drucker« an das Druckprogramm von 1st Mailmaster angepaßt.

Dort gelangt man endlich auch zu den Serienbrieffunktionen. Neben dem Serienausdruck kann man einen Probedruck vornehmen (der Brief wird einmal mit dem ersten Datensatz gedruckt) oder eine Serie von Etiketten mit den Textelementen des ersten Datensatzes drucken lassen. Pro Serienbrief sind bis zu neun verschiedene Textbausteine einbindbar. Im Text muß an den Einfügestellen der Klammeraffe (@), gefolgt von einer Ziffer zwischen 1 und 9 eingetragen werden. Ferner sind einige weiterere Steuerfunktionen implementiert wie zum Beispiel die Seitennummerausgabe an jeder beliebigen Stelle im Text.

Mehr ist eigentlich nicht zu beachten, um erfolgreich Serienbriefe zu erstellen. Das Programm funktioniert wie im Handbuch beschrieben. Die Bedienung ist mausgerecht gestaltet. Ein paar kleine Unzulänglichkeiten sind angesprochen worden. 1st Mailmaster fehlt sicherlich der letzte professionelle Schliff. Diese Beurteilung wird noch erhärtet durch die mehrfachen Warnungen im Handbuch vor der Gefahr des Datenverlustes durch unbeabsichtigtes Überschreiben von wichtigen Dateien. Besser als jede Mahnung wäre eine Sicherheitsabfrage.

Gar gräßlich hat er mal wieder zugeschlagen, der Rotstift unsres Chefkorrektors! Da hat man sich als armer Schreiberling alle erdenkliche Mühe



1st Lektor – lernbegieriges Wörterbuch

gegeben, auch das letzte Manuskript sauber ausgedruckt und ohne Eselsohren nur sieben Tage nach Redaktionsschluß bei den Redaktionsoberen abzuliefern. Und was ist das niederschmetternde Ergebnis? Dick und fett und rot geschrieben steht unter dem Produkt nächtelanger Tipparbeit: »Siebzehn Rechtschreibfehler, beim nächstenmal Duden benutzen!!«. Mit schamroten Ohren, viel roter noch als alle Korrekturzeichen, beugt man sich über sein Machwerk und stellt fast erleichtert fest, daß man eigentlich noch gut davongekommen ist. Denn der eine Fehler auf Seite 3 und die zwei »dicken Hunde« in der Bildunterschrift sind unentdeckt geblieben. Das muß beim nächsten Mal anders werden.

Keine Chance für Fehlerteufel

Und es wird anders! Selbst auf dem ach so arbeitsaufwendigen und konzentrationsfordernden Feld der Textkorrektur kann der gute Freund ST auf dem Schreibtisch hilfreich in die Bresche springen. Unter der Bezeichnung »1st Lektor« hat Atari ein Programm auf den Markt gebracht, das seinem Namen alle Ehre macht. »1st Lektor« ist zunächst einmal ein ganz vorzüglicher Rechtschreibkorrektor für deutschsprachige Texte. Er wird mit einem Standardwörterbuch (der Ordner »LEKTOR.WBU« auf der Programmdiskette) ausgeliefert, das augenblicklich mehr als 45000 Wörter umfaßt. Eine umfangreiche Erweiterung dieses Wörterbuches ist für die nahe Zukunft vorgesehen.

Nach Angaben des mitgelieferten Handbuches beträgt die programmtechnisch mögliche Maximalgröße 348000 Wörter. Darüber hinaus ist »1st Lektor« lernfähig. Bei der Korrekturarbeit können bis zu vier eigene Spezialwörterbücher erzeugt und für weitere Korrekturen herangezogen werden. Da jedes dieser Spezialwörterbücher bis zu 34800 Wörter enthalten kann, wäre es theoretisch möglich, Texte von einem Wörterbuch mit 487200 Wörtern durchsehen zu lassen (Der weitverbreitete Rechtschreibduden enthält etwa 180000 Stichwörter). Trotz Datenkompaktierung auf den Speichermedien würde dieses Monsterbuch ungefähr 1,2 Megabyte Speicherplatz einnehmen.

Wie arbeitet unser Superprogramm? Nach Anklicken von »LEKTOR.PRG« meldet sich »1st Lektor« mit einer etwas verspielten, aber durchaus zweckmäßigen grafischen Bedieneroberfläche. Die Steuerung erfolgt durch Mausklicken in die fünf Auswahlknöpfe am linken Bildschirmrand und die diversen Dialogboxen auf den beiden abgebildeten Buchseiten.

Zu Beginn einer Korrektursitzung wird der Text eingelesen. Dabei werden nur die verschiedenen Wörter erfaßt und alphabetisch sortiert, wobei aber auch unterschiedliche grammatikalische Formen desselben Wortes als verschieden betrachtet werden. Auf diese Weise läßt sich die Anzahl der zu überprüfenden Wörter erheblich reduzieren. Ein Mustertext für Zeittests mit »1st Lektor« mit 1604 Wörtern mit insgesamt 12281 Anschlägen enthielt knapp 780 verschiedene Wörter und Wortformen. Der Computerlektor benötigte zur Erfassung und Sortierung des Beispieltextes mit der Harddisk 42,6 Sekunden (RAM-Disk: 42,0 Sekunden, Floppy: 47,8 Sekunden).

Der nächste Arbeitsvorgang besteht in einem häppchenweisen Vergleich der aussortierten Wörter mit den Wörterbüchern. Findet »1st Lektor« mehr als zwanzig falsche Wörter, sucht er weiter bis zum Ende der Liste. Sind mehr als 150 Wörter falsch, unterbricht er schon vorher den Korrekturlauf und bietet die gefundenen unbekannten Wörter zur Überprüfung durch den Bediener und zur Übernahme in die Spezialwörterbücher an. Danach beginnt er auf Wunsch erneut mit der Korrektur. Bei weniger als zwanzig Fehlern zeigt er auf der linken Buchseite den Originaltext mit fett dargestellten Fehlerworten, auf der rechten Seite die Korrekturvorschläge. Es soll an dieser Stelle betont werden, daß »1st Lektor« nicht selbständig korrigiert, sondern nur Vorschläge macht, die der Benutzer übergehen, verändern oder anerkennen kann.

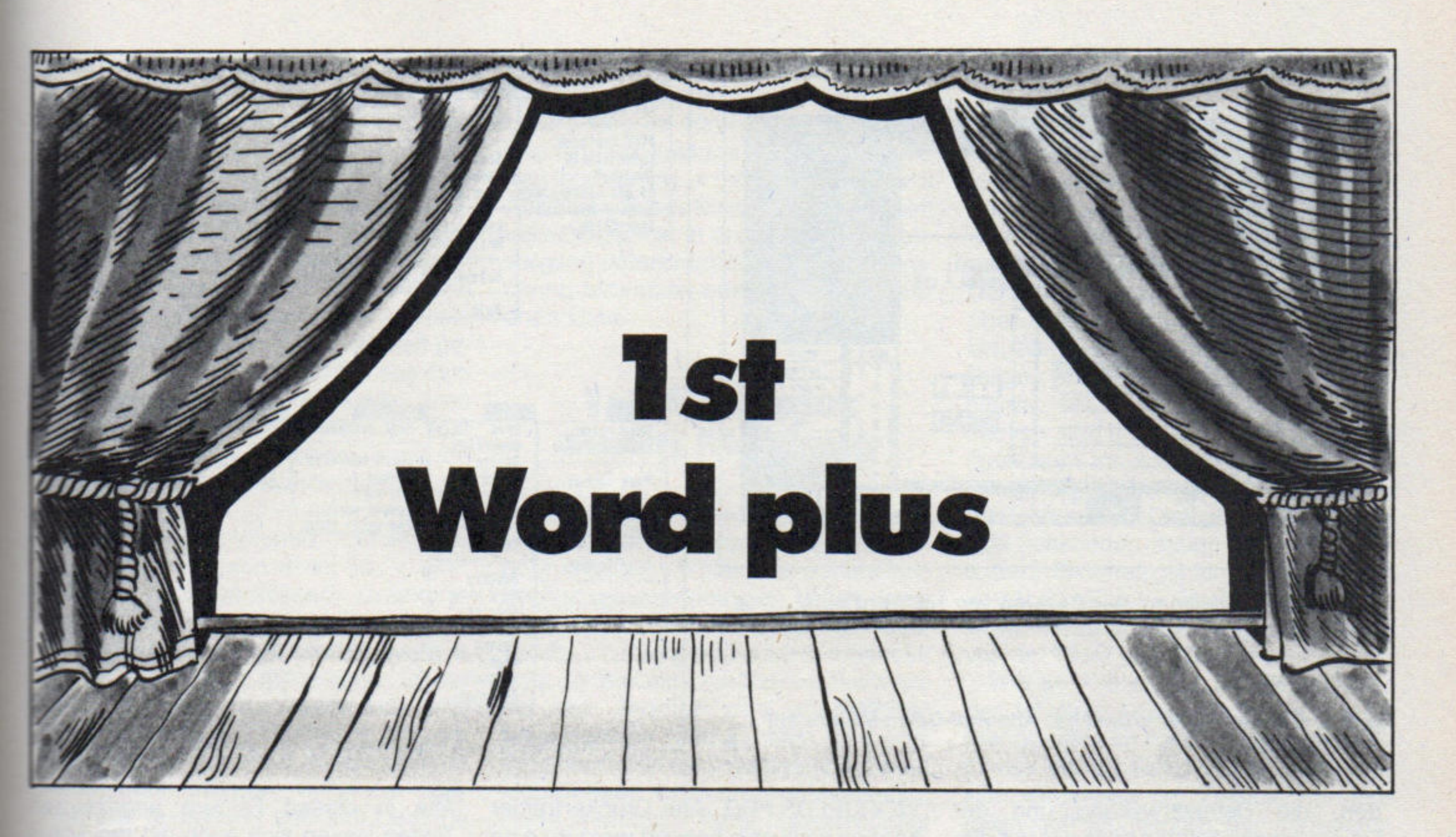
Ein kompletter Korrekturlauf mit unserem vorkorrigierten Mustertext (er enthielt nur noch ein falsches Wort) dauerte mit Texterfassung und Sortierung auf der RAM-Disk 6 Minuten und 50 Sekunden, auf der Harddisk 7 Minuten und 13 Sekunden und auf der Floppy-Disk 8 Minuten und 47 Sekunden. Die soeben beschriebene Programmphilosophie erlaubt auch die Korrektur wesentlich längerer Texte. Die Liste der verschiedenen Wörter, die ja allein Gegenstand der Untersuchung ist, wächst natürlich wesentlich langsamer als die Textlänge. Die Liste kann maximal 2400 Wörter aufnehmen. Sollte bei besonders großen Texten der vorgesehene Platz dennoch nicht ausreichen, nimmt »1st Lektor« den Umweg über externe Massenspeicher, indem er die überzähligen Wörter in Disk-Dateien ablegt und anschließend diese Dateien weiter korrigiert.

Neben der Textkorrektur besitzt »1st Lektor« noch hervorragende Fähigkeiten als Textanalysator. Von der Buchstabenverteilung über eine Wortlängen- und Worthäufigkeitsstatistik bis zu einer Berechnung eines sogenannten Text-Levels läßt sich alles in Linien- und Balkengrafik oder auch als Liste darstellen. Die Abbildungen vermitteln eine gute Vorstellung von den angebotenen Darstellungen.

»1st Lektor« ist ein hervorragendes Arbeitsmittel für Vielschreiber unter den »Schreibtischtätern«. Plötzlich macht Textkorrektur sogar richtig Spaß. Der Rotstift des Chefkorrektors wird arbeitslos. (W. Fastenrath/hb)

Info: LQ-Font, Computer-Technik Kieckbusch, 5412 Ransbach, Am Seeufer 22, Preis: 149 Mark
1st Mailmaster, Computare oHG, 1000 Berlin 30, Keithstr. 20, Preis: 99 Mark
1st Spooler, Atari Corp. Deutschland GmbH, 6096 Raunheim, Preis 99 Mark
1st Lektor, Atari Corp. Deutschland GmbH, 6096 Raunheim, Preis: 149 Mark





1st Word plus

Das brandneue 1st Word plus vereint Text und Grafik. Kombiniert mit einfacher Bedienung und leistungsfähigen Funktionen setzt es neue Maßstäbe für alle Textverarbeitungen.

Eineinhalb Jahre ist es nun her, seitdem das Atari ST-System das Licht des deutschen Computermarktes erblickt hat. Seitdem sind schon viele Bits durch die Leiterbahnen der diversen STs gejagt worden, der ursprüngliche 520 ST hat inzwischen nicht weniger als drei neue Geschwister bekommen. Auf manch privilegiertem Arbeitsplatz soll sogar schon die lang erwartete Festplatte sanft vor sich hin röhren. Software für die verschiedensten Anwendungsbereiche gibt es in großer Vielfalt und Qualität, nur auf die angekündigte Softwaregrundausstattung GEM-Paint und GEM-Write wartet die geduldige ST-Gemeinde immer noch vergeblich. Dabei stellte doch die versprochene einfache Einbindung von GEM-Paint-Bildern in die GEM-Write-Texte ein wichtiges Entscheidungskriterium für den Kauf eines Atari-ST-Systems dar.

Lobend anzuerkennen ist jedoch die Tatsache, daß die Herren aus Raunheim nach anfänglichem Zögern relativ rasch auf das unüberhörbare Murren der ST-Benutzer reagiert haben und, zumindest bei der reinen Textverarbeitung, für mehr als guten Ersatz gesorgt haben. Zum Jahreswechsel 1985/86 brachte Atari als Weihnachtsüberraschung das wirklich hervorragende Textverarbeitungsprogramm 1st Word (deutsch: Das Erste Wort) der englischen Firma GST auf den Markt (Test in Happy-Computer 2/86, Seite 104). Dieses Programm wurde zwischenzeitlich mehrfach verbessert und ist in der Version 1.06 auch mit deutscher Bedienerführung erhältlich. Aufgrund seiner weiten Verbreitung unter den ST-Besitzern kann man es mit Fug und Recht als Textverarbeitungsstandard für ST-Computer bezeichnen.

Geschichte eines Standards

Schon die ersten Versionen von 1st Word waren geradezu Musterbeispiele für eine gelungene Einbindung von Programmen in die grafische Bedieneroberfläche GEM. Kaum ein Benutzer hat je das als Diskettendatei mitgelieferte Handbuch zu Rate ziehen müssen, um die vielfältigen Funktionen des Programmes sicher zu beherrschen. Dem logischen Aufbau der Pull-Down-Menüs und der wohldurchdachten Gestaltung des GEM-Desktop, mit Textdarstellung in vier voneinander unabhängigen Fenstern, konnten selbst ausgesprochene Maushasser ihre Anerkennung nicht verweigern.

Bei allem Lob für das wohldurchdachte Konzept muß man den Programmierern aber dennoch vorwerfen, daß einige wichtige Kleinigkeiten nicht implementiert waren und daß bei einigen Funktionen wie zum Beispiel dem Verwalten und Ausdrucken von Textdateien Programmabläufe gewählt wurden, die dem ST als einen Computer mit besonders großem Direktzugriffsspeicher nicht gerecht werden konnten. So löschten die bisherigen Versionen von 1st Word die Texte nach dem Speichern vom Desktop, das Drucken der Texte über ein nachzuladendes Druckprogramm war nur bei leerem Desktop möglich. Das Einbinden von grafischen Darstellungen in die Texte war ohnehin nicht vorgesehen. Das konnte unmöglich das letzte Wort aus England sein!

Die CeBIT '86 bestätigte diese Vermutung! Auf dem Atari-Stand konnte man die erste Vorversion eines Programmes bewundern, das durch seinen Namen »1st Word plus« eine Erweiterung des alten guten Standards versprach. Grafik konnte zwar auf dem Bildschirm in Texte integriert werden, beim Ausdrucken war jedoch von den schönen Bildern, außer einem Stück weißen Papieres in Größe der Grafik, nichts zu sehen. Die schon angesprochenen fehlenden Kleinigkeiten, wie etwa die Anzeige der Cursorposition im Text, waren immer noch nicht vorhan-

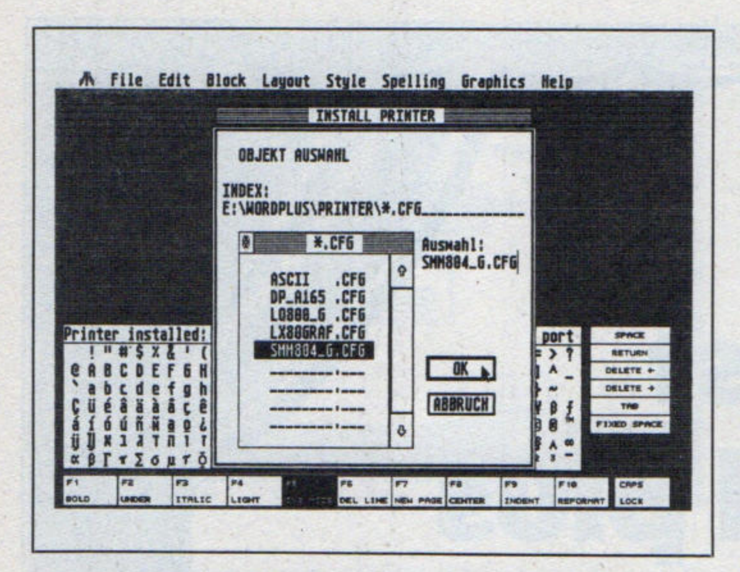


Bild 1. Klare Aussagen: Der Desktop von 1st Word plus ▲

1st Word Plus... · VT52 Emulator · Kontrollfeld · Snapshot · RS 232 Einstellung · Drucker Anpassung · Side-Click

File: Open file... · Print file... · Save file · Save as... · Save and resume · Read file... · Write block... · Delete file... · Quit

Edit: ✓ WP mode · ✓ Insert mode · Statistics... · Find... · Replace... · Repeat find · Set mark... · Goto mark... · Goto page...

Block: Start block · End block · Cut block · Paste block · Copy block · Move block · Delete block · Find start · Find end · Hide block

Layout: Show ruler · ✓ Show position · Page layout... · Add ruler... · Delete ruler · Read format... · Footnote...

Style: Bold · Underline · Italic · Light · Superscript · Subscript · Restyle · Justify · ✓ Word wrap · Spacing · Center · Right · Indent · Reformat...

Spelling: Load dictionary · Check spelling · Browse... · Add word... · Remove word... · End spell check

Graphics: ✓ Graphics mode · Read picture... · Delete picture

Help: Extra help · Editing · Layout · Margins · Tab points · Typing · Correcting · Cursor · Scrolling · Deletion · Keyboard · Page breaks · Cut & paste · Printing

Bild 2. Am Ziel vieler Wünsche: Alle Pull-Down-Menüs auf einen Blick ▶

den, die Dateiverwaltung und der Druckvorgang zeigten nach wie vor dieselben seltsamen Abläufe. Also doch erst das vorletzte Wort?! Aber immerhin, Hoffnungen waren geweckt, und Atari konnte sich fortan über Mangel an Fragen nach dem Wie, Wann und Wo von 1st Word plus nicht beklagen. Die hartnäckigen Anfragen der Fachjournalisten entlockten der Raunheimer Mannschaft bald nur noch leicht gequälte Seufzer.

Neuer Schreibtisch

Besonders Neugierige und eine hohe Telefonrechnung nicht scheuende ST-Besitzer brauchen die unermüdliche Schaffensfreude der deutschen Atari-Zentrale nun nicht länger zu trüben. Es ist uns gelungen, die neueste Version von 1st Word plus direkt aus England herbeizuschaffen, um sie für unsere Leser unter die Maus zu nehmen.

In den folgenden Ausführungen werden Funktionen, die im Vergleich zum »alten« 1st Word (ohne plus) unverändert geblieben sind, nur kurz erwähnt, besonderes Gewicht soll auf die Verbesserungen und Erweiterungen gelegt werden. Auf der Programmdiskette befinden sich ein Ordner PRINTER mit einigen Druckertreibern und dem zugehörigen Installationsprogramm INSTALL.PRG und die Dateien PRINTER.CFG, SNAPSHOT.ACC, SPELLING.DIC, WORDHELP.RSC, WORDMSGS.RSC, WORDPLUS.RSC und WORDPLUS.PRG. Alte Druckertreiber können weiterhin benutzt werden, eine Neuinstallation mit INSTALL.PRG ist allerdings erforderlich. Die erzeugten Druckertreiber besitzen die Dateibezeichnung ».CFG«. 1st Word plus startet man durch Doppelklick auf WORDPLUS.PRG. Dabei wird auch ein Druckertreiber mit der Bezeichnung »PRINTER.CFG« mitgeladen. Auf dem Bildschirm erscheint ein Desktop, ähnlich dem bekannten 1st Word.

Trotz der erhöhten Anzahl der Programmfunktionen konnte der Bedienungskomfort noch gesteigert werden. Die bekannte Tabelle der darstellbaren und über Mausklick in den Text integrierbaren Zeichen ist an eine günstigere Position gesetzt worden und hat einige Erweiterungen erfahren (Bild 1). So werden nur noch diejenigen Zeichen dargestellt, die der angeschlossene Drucker über den editierbaren Druckertreiber auch tatsächlich zu Papier bringen kann. Name und Anschlußport des Druckers sind in einer Titelzeile angegeben. Nach Anklicken des Druckernamens kann man aus einer Dateiauswahlbox andere Druckertreiber laden. Rechts neben dieser Tabelle befinden sich sechs Felder, die nach Anklicken an der Cursorposition zwei verschiedene Leerstellen, Zeilenvorschübe, Tabulatorsprünge oder Buchstabenlöschungen bewirken. Am unteren Bildschirmrand gibt es neben der Leiste mit der Funktionstastenbelegung ein Feld, das den aktuellen Schaltzustand der CAPS LOCK-Taste anzeigt. Alle in diesen Feldern angezeigten Tasten lassen sich auch mit der Maus bedienen.

Erhebliche Verbesserungen sind im Bereich der Bildschirmsteuerung zu vermelden. Die Kaffeepausen beim Scrollen größerer Textdateien von Textanfang zu Textende müssen leider künftig entfallen. Auch das Nachlaufen des Cursors bei Steuerung durch die Tastatur gehört der Vergangenheit an. Unaufmerksame 1st-Word-Schreiber haben manches Wort oder gar manchen Satz neu schreiben müssen, wenn sie bei schneller Tastaturwiederholungsrate zu lange auf die »DELETE«-Taste gedrückt hatten. Im 1st Word plus wird der Tastaturpuffer nach Freigabe der gedrückten Taste sofort gelöscht.

6 + 3 = plus

Die verfügbaren Pull-Down-Menüs mit ihren Einzelfunktionen sind in Bild 2 wiedergegeben. Der 1st-Word-Kenner sieht mit einem Blick, daß die Menüleiste außer den bekannten sechs Menüs drei neue, nämlich »Layout«, »Spelling« und »Graphics« enthält. Doch nicht nur hier hat sich etwas getan, auch hinter einigen altbekannten Menüpunkten verbergen sich wichtige Neuerungen.

Im Menü »Atarizeichen« befinden sich wie gewohnt die Copyright-Meldung und die Desktop-Accessories. Unter diesen Accessories stößt man auf eine neue mit Namen »Snapshot«,

die zum Lieferumfang gehört. Mit Hilfe von »Snapshot« werden grafische Darstellungen zur Einbindung in den Text auf Diskette oder Festplatte gespeichert. Genaueres hierüber erfahren Sie bei der Besprechung des Menüs »Graphics«. Denn wir wollen in der Reihenfolge der Pull-Down-Menüs vorgehen. Im Menü »File« ist der Punkt »Save and resume« ergänzt worden. Unter diesem Menüpunkt kann der Text jederzeit gespeichert werden, ohne daß er aus dem Computerspeicher gelöscht wird. Man kann also während der Arbeit am Text nach Herzenslust speichern, ohne anschließend den Text erneut laden zu müssen. Endlich!

Die Druckfunktionen, erreichbar durch Anklicken von »Print file...«, wurden erheblich verbessert. Druckprogramm und Druckertreiber muß man nicht mehr nachladen, sondern werden bei Programmstart mitgeladen. In der Druckparameter-Box lassen sich jetzt auch dreistellige Offsetwerte für die Seitenzahlen eingeben. Ausdrucken ist selbst dann möglich, wenn ein oder mehrere Textfenster auf dem Desktop geöffnet sind. Mehr noch, während des Druckens können andere Texte weiterbearbeitet werden: 1st Word plus druckt also im Hintergrund. Während des Druckvorganges ist der Menüeintrag in »Printing...« umgewandelt. Nach Anklicken dieses Menüfeldes erscheint eine Dialogbox, in der man den Druckvorgang unterbrechen, nach Unterbrechung fortsetzen oder gänzlich abbrechen kann.

1st Word plus macht Meldung

Die übrigen »File«-Funktionen entsprechen den früheren Versionen. Dennoch wurde auch hier eine weitere Verbesserung vorgenommen. Immer dann, wenn durch die Maus eine Dateiauswahlbox hervorgerufen wird, erscheint über der Box eine zusätzliche Textleiste mit der Anzeige des angewählten Menüpunktes (Bild 1). Man wird vom Programm her also wenigstens darauf hingewiesen, daß man gerade dabei ist, seine wichtigste Datei unbeabsichtigt zu löschen. Eine Sicherheitsabfrage haben die Programmierer leider nicht vorgesehen.

Das Menü »Edit« ist um zwei Funktionen erweitert worden. Einerseits lassen sich in einer Box statistische Daten über die gerade bearbeitete Textdatei und den freien Speicherplatz im RAM und auf den Massenspeichern abrufen (»Statistics...«), anderseits kann man über den Menüpunkt »Goto page...« in längeren Texten den Cursor an den Beginn einer bestimmten Seite positionieren. Mit »Set mark...« und »Goto mark...« sind unverändert vier beliebige Positionen im Text markierbar. Festlegung und Anspringen der Textmarker erfolgen in einer Dialogbox. Die übrigen Edit-Funktionen haben sich nicht verändert. Gleiches gilt für alle Funktionen im Menü »Block«. Unter diesem Pull-Down-Menü sind die Blockoperationen von 1st Word plus abrufbar.

Wie ein Auszug aus der Wunschliste der 1st-Word-Anwender lesen sich die Menüpunkte im neuen Menü »Layout«. In der ersten Fensterzeile kann man sich wahlweise die Zeilenlängen-Einteilung mit Tabulatoren, den sogenannten Ruler (»Show ruler«), oder Seite,

Zeile und Spalte der Cursorposition (»Show position«) anzeigen lassen. »Page layout« entspricht in seiner Funktion exakt der Layout-Funktion, die bei 1st Word noch aus dem File-Menü abgerufen wurde. Auch die Voreinstellung auf englische Papiermaße mit 66 Zeilen pro Blatt war zumindest in unserer Version unverändert beibehalten worden.

Zeilenwechsel à la carte

Die folgenden vier Menüpunkte sind jedoch echte Neuerungen. Anklicken von »Add ruler...« aktiviert eine Dialogbox (Bild 3), in der man eine neue Zeileneinteilung nach Zeilenlänge und Tabulatorweite bestimmen kann. Darüber hinaus ist auch die Änderung der Schrifttype für den Ausdruck wählbar. Dies erfordert jedoch eine entsprechende Anpassung des Druckertreibers. Vorgesehen sind die Typen Pica, Elite, Komprimiertschrift und Breitschrift. Schrifttypen, die wohl jeder Drucker beherrscht. Die Neufestlegung gilt immer ab der Zeile, in der sich der Cursor bei Anklicken von »Add ruler...« befindet. Sie behält ihre Gültigkeit bis zur nächsten Zeileneinteilung oder gar bis zum Textende. »Delete ruler« entfernt die Zeileneinteilung, die für die Textzeile mit dem Cursor gilt. Die vier erwähnten Schrifttypen können auch durch Anklicken des Typennamens am rechten Ende des Rulers umgeschaltet werden. Als weitere Neuerung ist das Festlegen des linken Textrandes zu erwähnen. Allerdings haben die Programmautoren bei dieser Funktion ein wenig gemogelt. Sie lassen nämlich das Programm entsprechend der Einrückung des linken Randes in jeden Absatz die passende Anzahl Leerstellen als Einrückung einfügen. Dadurch läßt sich eine solche Texteinrückung leider nicht einfach mit dem »Reformat...«-Befehl wieder rückgängig machen. Man muß vorher die Einrückung in der ersten Zeile eines jeden Absatzes durch Betätigen der »Delete«-Taste löschen.

Die beiden restlichen Funktionen des Layout-Menüs waren in unserer Version noch nicht implementiert. Nach Auskunft des Herstellers wird man mit »Read format...« vorbereitete Textseitenlayouts (zum Beispiel für Formbriefe) vom Massenspeicher einlesen können. Unter dem Menüpunkt »Footnote...« kann man in der Auslieferungsversion eine Fußnotenverwaltung ansprechen. Die Fußnoten werden durch hochgestellte Zahlen markiert sein, die das Programm automatisch durchnumeriert, auch bei späterer Einfügung von neuen Fußnoten. Das Anklicken einer Fußnotenmarkierung öffnet ein zusätzliches Fenster, in dem der Fußnotentext editiert werden kann.

Die Funktionen des Menü »Style« werden fast unverändert aus 1st Word übernommen. Die Textdarstellung ist jedoch nach Programmstart auf Flattersatz und nicht mehr wie bei 1st Word auf Blocksatz (»Justify«) voreingestellt. Durch Anklicken von »Reformat...« kann man nunmehr auch größere Textabschnitte bis hin zum gesamten Text mit einem Mausklick umformatieren. In der Endversion werden dabei auf Wunsch Trennvorschläge angegeben, die sich als sogenannte weiche Trennungszeichen in den Text einfügen lassen. Diese Trennzeichen erscheinen also nicht, wenn Worte mit weichen Trennzeichen bei eventuellen späteren Umformatierungen in die Mitte einer Zeile gelangen. Nach Auskunft des englischen Herstellers paßt das Programmierer-Team gerade die Trennvorschläge an die Trennungsregeln der deutschen Sprache an.

Die englischsprachige Version von 1st Word plus enthält einen einfachen Rechtschreibkorrektor (Pull-Down-Menü »Spelling«), einen sogenannten Speller, der neben der eigentlichen Rechtschreibkorrektur auch die Erweiterung des Wörterbuches zuläßt. Nach Auskunft der Softwareabteilung von Atari Deutschland ist jedoch nicht sicher, ob das deutsche 1st Word plus mit diesem Speller ausgeliefert wird. Atari favorisiert für diesen Zweck ein anderes Produkt aus deutscher Softwareproduktion mit dem Namen »1st Lektor«.

Bildgestaltung

Wer inzwischen ungeduldig geworden ist, weil noch immer nichts über die Einbindungen von grafischen Darstellungen gesagt wurde, braucht nun nicht mehr länger zu warten: Wir sind beim Menü »Graphics« angekommen! Das Anklicken des Menüpunktes »Graphics mode« schaltet die Textdarstellung auf dem Bildschirm zunächst einmal auf kleinere Schrifttypen um. Bild 4 zeigt die Bildschirm-Hardcopy eines Beispieltextes mit Grafikeinbindung. Die kleinere Schrifttype entspricht ungefähr der Buchstabengröße auf dem ausgedruckten Blatt. Daher kann man schon auf dem Bildschirm das endgültige Aussehen eines Ausdrucks mit Grafik und Text beurteilen. Durch Anklicken von »Read picture...« lassen sich nun Bilder von Diskette oder Festplatte an der Cursorposition einlesen. Bilder werden vor der Cursorzeile eingefügt, die Cursorzeile und die nachfolgenden Textzeilen entsprechend der

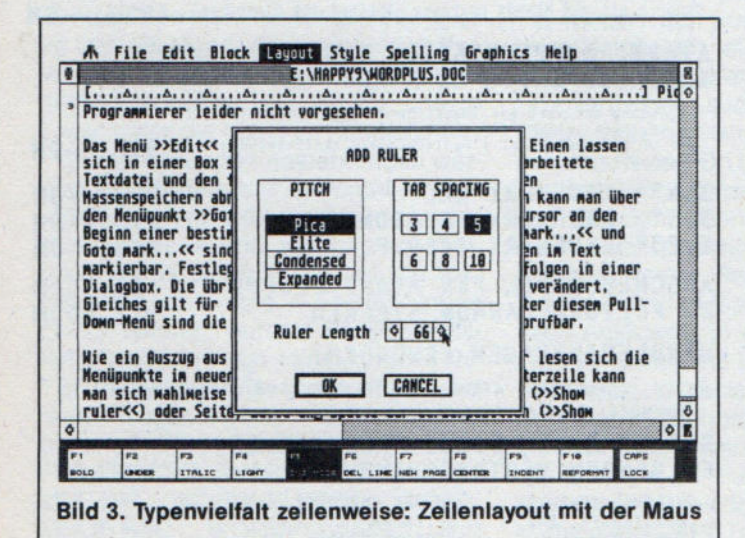


Bild 3. Typenvielfalt zeilenweise: Zeilenlayout mit der Maus

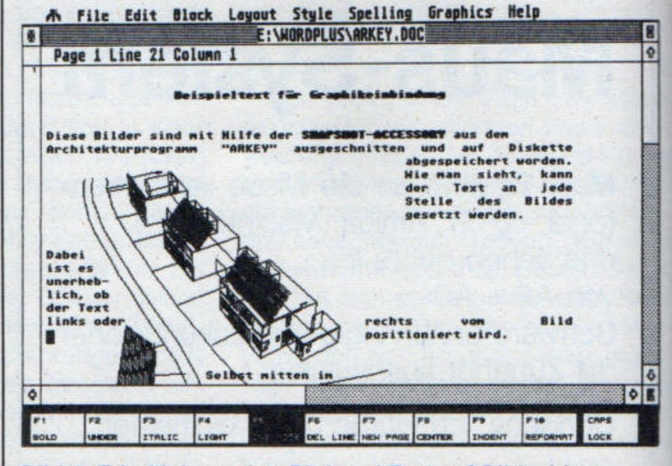


Bild 4. Friedlich vereint: Bild und Text auf Bildschirm

Bildgröße zeilenweise nach hinten verschoben. Durch die normalen Text- und Blockoperationen des Programms kann der Text an die gewünschten Stellen vor, hinter, neben oder mitten in das Bild geschrieben werden. Das Bild selbst ist durch Anklicken und Festhalten der linken Maustaste auf dem Bildschirm frei verschiebbar. Ein Mausklick auf »Delete picture« entfernt ein Bild wieder aus dem Text, der Text, auch wenn er im Bild positioniert war, beeinflußt diese Operation nicht.

Die Programmierer von 1st Word plus haben einen Weg gefunden, Grafiken von nahezu allen auf ST-Computern laufenden Zeichen- und Malprogrammen für die Einbindung verfügbar zu machen, ohne auf die verschiedenen Dateiformate der Grafikprogramme Rücksicht nehmen zu müssen. Alle Grafikprogramme haben natürlich zwangsläufig eine Funktion gemeinsam: Sie stellen ihre Bilder auf dem Bildschirm dar. Bei dieser an sich trivialen Erkenntnis haben die Programmautoren angesetzt und ein Accessory programmiert, nämlich das schon kurz erwähnte Programm »SNAPSHOT.ACC« im Pull-Down-Menü »Atarizeichen«.

Diese Accessory ist in der Lage, aus jeder Bildschirmdarstellung einen rechteckigen Teil oder sogar den gesamten Bildschirm mit Hilfe des bekannten GEM-Gummifadens auszuschneiden und in drei verschiedenen Formaten auf Diskette oder Festplatte zu speichern. Darunter befindet sich auch ein spezielles, für 1st Word plus lesbares Format, das die grafischen Dateien in kompaktierter Form speichert. Derartige Bildausschnitte lassen sich mit anderen Grafikprogrammen wie dem einfachen »Doodle« oder bei Farbgrafiken mit »Neochrom« nachbearbeiten. Die einzige Voraussetzung für die Anwendung dieser beinahe genialen Idee ist die Einbindung der Grafikprogramme in eine GEM-Oberfläche, die das Aufrufen von Desktop-Accessories erlaubt.

Dies ist leider bei einigen besonders beliebten und weitverbreiteten Programmen wie zum Beispiel »Degas« nicht der Fall. Da man aber mit Hilfe von Formatwandelprogrammen die meisten Grafiken aus Programmen ohne GEM-Einbindung für »Doodle« lesbar machen kann, wird es auch mit derartigen Grafiken keine grundsätzlichen Probleme geben.

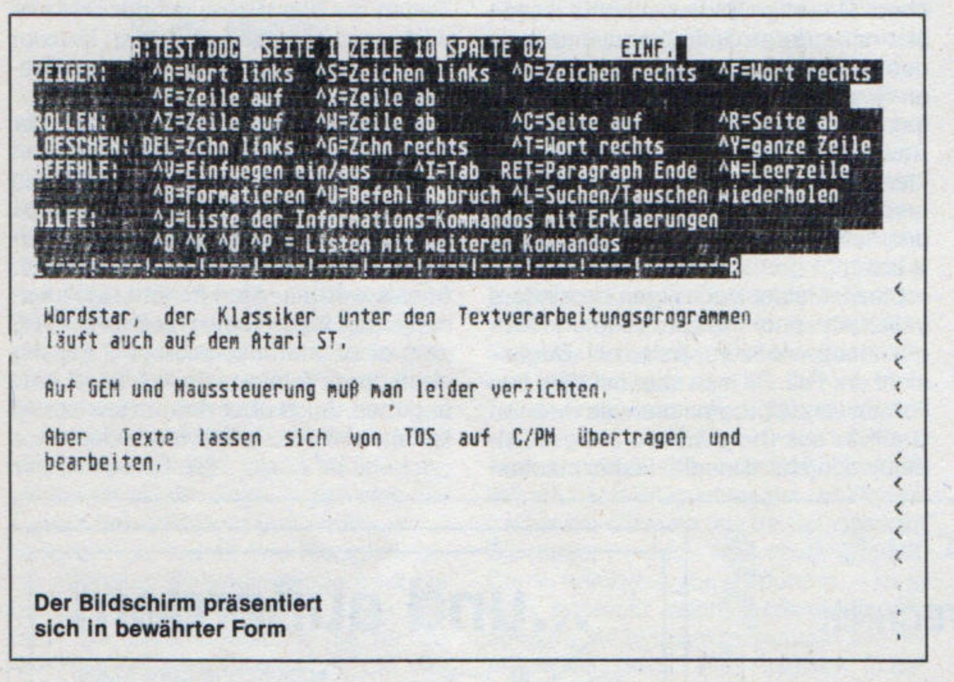
Die Bildinformationen zur Einbindung in 1st-Word-plus-Texte werden nicht in die normalen Textdateien gespeichert. Dort befinden sich lediglich die Informationen über den Namen der einzubindenden Bilddatei und über die Position des Bildes im Text. Der Anwender muß also darauf bedacht sein, daß beim Laden und Drucken von bebilderten Texten die Bilddateien auf der Diskette vorliegen. Ein Fehlen dieser Dateien quittiert 1st Word plus mit einer Fehlermeldung.

Wer jetzt schon in Gedanken die Inhalte seiner Disketten mit den vielen schönen Grafik-Demos durchgeht, um den nächsten Brief an Tante Frieda mit hübschen Bildern zu verzieren, muß leider an dieser Stelle noch ein wenig vertröstet werden. Atari konnte noch keinen endgültigen Termin für die Auslieferung einer internationalen oder gar der deutschen Version von 1st Word plus angeben. Auch über den Verkaufspreis ließ man nichts Definitives verlauten.

(W. Fastenrath/hb)

Wordstar: Die Legende lebt

Wordstar ist ein Evergreen unter den Textverarbeitungsprogrammen; altbewährt und weiterhin gut verrichtet es seinen Dienst auf dem ST.



Der Bildschirm präsentiert sich in bewährter Form

Es war eine Sensation, als die Programme dBase, Wordstar und Multiplan auch für die Heimcomputer Schneider CPC und Commodore 128 zum Preis von knapp 200 Mark auf den Markt kamen. Endlich konnte man für wenig Geld echte Personal-Computer-Luft schnuppern. Zur gleichen Zeit aber machte ein neuer Stern am Computerhimmel Furore, der Atari ST. Mancher unentschlossene Computerkäufer stand vor dem Dilemma: Soll man sich für einen schon etwas bemoosten 8-Bit-Computer entscheiden, aber dafür echte CP/M-Power genießen, oder in die noch risikoreiche Welt von GEM und 68000-Prozessor einsteigen?

Wie eine Erlösung war es für viele, als die Ankündigung laut wurde, daß auch die Anhänger der Atari-Gemeinde in den Genuß von CP/M-Software kommen können, und das ebenfalls preisgünstig. Innerhalb kürzester Zeit war dann auch das bekannte Textsystem Wordstar 3.0 mit seinem Mailmerge erhältlich. Sicherlich unken einige, daß dieses Programm ein absoluter Oldie ist und eigentlich den ST in seinen Fähigkeiten zu einem erheblichen Understatement zwingt. Um es vorweg zu sagen: Wer viel Wert auf Unterstützung durch GEM legt, der ist bestimmt mit dem Programm 1st Word von GST besser bedient. Doch auch die Arbeit mit Wordstar ist einem ST-Besitzer von großem Nutzen, vor allem dann, wenn hinter der Hobbyanwendung berufliche Ziele stehen. Denn die Erfahrung zeigt: Wer mit Wordstar umgehen kann, der sitzt auch bei anderen Textsystemen sicher im Sattel. Da Wordstar unter den Textverarbeitungen die wirklich erste gute war, orientierten sich viele Programmierer daran, so daß es fast so was wie einen »Wordstar«-Standard gibt.

Nun zu der praktischen Arbeit mit diesem Programm. Wordstar wird auf zwei Disketten mit den Originalhandbüchern geliefert. Diese sind sehr umfangreich und aus diesem Grunde für den Einsteiger etwas unübersichtlich. Der Kauf eines Begleitbuches ist deshalb unbedingt zu empfehlen, besonders wenn man schnell alle Fähigkeiten nutzen möchte. Der CP/M-Emulator, den das Programm beim ST voraussetzt, ist übrigens ebenfalls im Lieferumfang enthalten.

Nach dem Start meldet sich Wordstar mit einem umfangreichen Hauptmenü. Alle Meldungen erscheinen in Deutsch. Ein ständig sichtbares, auf Wunsch abschaltbares, Inhaltsverzeichnis gibt Auskunft über die bereits vorhandenen Dateien. Leider schweigt es sich bezüglich der Länge der einzelnen Datensätze sowie über den noch vorhandenen Speicherplatz vollkommen aus. Dieser ist durch das CP/M ohnehin auf rund 116 KByte pro Diskette beschränkt. Der Besitz einer SF 314 bringt da auch keinen Speicherplatzvorteil, da CP/M nur das einseitige Beschreiben einer Diskette gestattet. Das Anlegen einer reinen Textdiskette empfiehlt sich also, vor allem für Vielschreiber.

Nach der Einstellung einer der vier Hilfsstufen gelangt man vom Hauptmenü in den Eingabemodus. Wordstar erlaubt an dieser Stelle sowohl die Einrichtung einer Textdatei als auch die einer reinen ASCII-Datei für Programmdateien. Auf diese Weise besitzt man nicht nur ein Programm für Briefe und sonstige Texte, sondern auch gleich einen hervorragenden Editor ohne rechte und linke Randbegrenzung für das Schreiben von Programmen verschiedener Sprachen.

Es ist wirklich ein Vergnügen, mit diesem System seine anfallende Schreibarbeit zu erledigen. Über ein ständig sichtbares Lineal legt man beide Zeilenränder fest, die selbstverständlich jederzeit wieder zu ändern sind. Wird ein Text nach Vollendung umformatiert, so macht Wordstar auch Trennvorschläge. Über die Cursorsteuertasten ist jede Textstelle sofort erreichbar. Leider hinkt bei Benützung der Pfeil- und der Delete-Taste die Repeatfunktion erheblich nach. So fährt der Cursor auch nach dem Loslassen der betreffenden Taste noch munter durch die Gegend. Vor allem bei der DEL-Taste führt dies mitunter zu unfreiwilligen Kürzungen. Also: nicht zu lange auf diesen Tasten verweilen. Die Eingabe selbst jedoch gestaltet sich hervorragend. Jede Zeile wird sofort auf die gewünschte Zeilenbreite, in Block- oder Flattersatz formatiert. Mit »Word Wrap« wird jedes Wort, das nicht mehr in ganzer Länge in die Zeile paßt, automatisch in die nächste gezogen. Was Blindschreiben vereinfacht. Die Verwendung der Return-Taste ist nur nach einem Absatz erforderlich. Der Text erscheint auf dem Schirm exakt so, wie er später auf dem Papier aussehen wird.

Ständige Ausdrucke sind im Hinblick auf den hervorragenden Monitor von Atari vollkommen überflüssig. Zeilenlängen über 80 Zeichen (bis maximal 255) bewirken ein Scrolling des gesamten Bildschirms, was der Übersichtlichkeit bei breiten Tabellen sehr dienlich ist. Wordstar erlaubt ganze Textstücke als Blöcke zu definieren, die kopiert, verschoben, gelöscht oder sogar in andere Dateien überführt werden können. So speichert man besonders oft verwendete Formulierungen einfach separat ab und verwendet sie innerhalb von anderen Texten beliebig oft. Die Texte werden wahlweise mit rechtem Randausgleich ausgedruckt, was eine zeitungsähnliche Textdarstellung bewirkt. Wörter können Sie unterstreichen, durchstreichen und auch hoch oder tiefer stellen. Wichtige Textstellen werden mit Doppel- oder Dreifachanschlag besonders hervorgehoben. Bei mehreren Seiten kann man die Ausgabe der Seitennummer und eventueller Kopf- oder Fußzeilen vorgeben. Wordstar unterstützt nach Angabe des Herstellers alle gängigen Drucker mit Centronics-Schnittstelle.

Leider muß man, um alle Fähigkeiten von Wordstar in den Griff zu bekommen, ein wahrer Akrobat auf der Controltaste sein. Diese Taste ist das A und O jeglicher Kommandoeingabe und erfordert einige Gewöhnung. Da schaut so mancher ST-Anwender sehnsüchtig auf die arbeitslose Maus neben seinem Computer. Glücklicherweise hat MicroPro eine Referenzkarte mitgeliefert, die bei den ersten Gehversuchen unbedingt neben das Terminal gehört. Steht ein Indexzeichen vor einem Buchstaben, so muß die Controltaste verwendet werden. Kommandos mit zwei Buchstaben erfordern eine gleichzeitige Eingabe von Control und des ersten Zeichens, um in das entsprechende Untermenü zu gelangen. Das zweite Zeichen wählt einen Punkt daraus an.

Auch die Funktionstasten sind belegbar. So löst man wenigstens die oft benutzten Kommandos mit nur einem Tastendruck aus. Erscheint dies alles auf den ersten Blick sehr kompliziert, geht es jedoch schon nach kurzer Zeit in Fleisch und Blut über. Textverarbeitungen mit GEM sind zugegebenermaßen in dieser Beziehung angenehmer zu bedienen.

Zu den besonderen Leckerbissen des Paketes gehört die MailMerge-Funktion, die ein in sich geschlossenes Programm darstellt. Möchte man seinen Freunden zu irgendeinem Anlaß ein Einladungsschreiben schicken, ist es mit einem einzelnen Schreiben nicht getan. Sicher, wem es auf Zeit nicht ankommt, der kann auch 20mal denselben Text mit unterschiedlicher Anrede niederschreiben. Stolze Besitzer von Wordstar jedoch verfassen Ihre Einladung nur einmal. Variable Daten wie Adresse, Anrede oder Grußformel werden in einer Extra-Datei abgelegt und fließen selbständig in die entsprechenden Texte ein. Zu guter Letzt erlaubt MailMerge noch den Ausdruck der passenden Adreßaufkleber. Benutzer eines dBase-II-Paketes können außerdem völlig problemlos Adreßdateien in ihre Texte übertragen.

Wordstar ist ein ideales System, um den Umgang mit Profitextsystemen zu erlernen. Wer jedoch Wert auf die grafischen Fähigkeiten seines Ataris legt, gibt besser einem Programm wie 1st Word von GST den Vorzug. Es bleibt aber zu wünschen, daß die Idee, CP/M-Software für den Atari so preisgünstig auf den Markt zu bringen, noch öfter aufgegriffen wird.

(Peter Herrmann/hb)

Pluspunkte für »ST-Pascal plus«

Stößt bald schon ein neues Pascal, das »ST Pascal Plus«, C als Königssprache vom Thron? Es ist einfach, gut dokumentiert und preiswert.

Das deutsche Softwarehaus CCD machte sich bereits mit dem Pascal-Compiler »ST-Pascal« einen guten Namen. Jetzt brachte es eine überarbeitete und erweiterte Version unter dem Namen »ST-Pascal plus« auf den Markt. Das Programmpaket für 249 Mark vertreibt sowohl CCD als auch, wie schon die Vorgängerversion, Atari selbst.

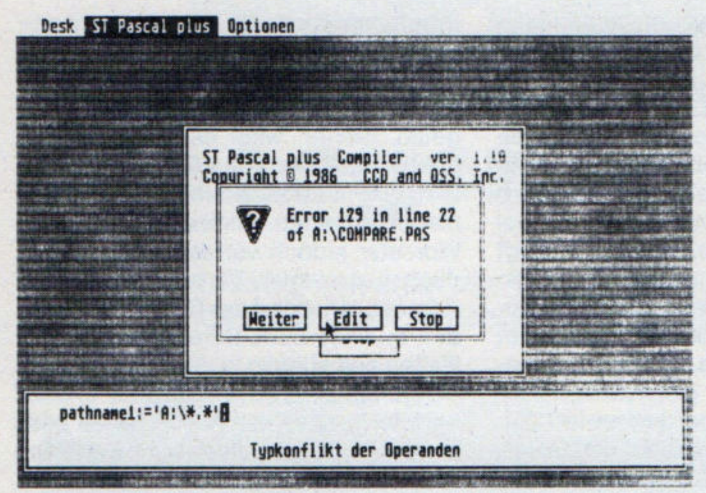
Bei CCD nahm man sich die Kritik zu einigen Schwachstellen der bisherigen Version zu Herzen. Dazu zählen das etwas knappe Handbuch, der Programmschutz des Compilers und die umständliche Bedienung des Systems.

Für den Programmschutz legte man nun eine für alle Seiten akzeptable Lösung vor: Die Systemdiskette ist beim Kauf kopiergeschützt. Sie verwandelt sich jedoch bei Eingabe des Namens des Käufers und der Seriennummer in eine ungeschützte Diskette, die sich gegen das Kopieren – natürlich nur für den privaten Gebrauch – nicht mehr sträubt. Das Anlegen einer Sicherheitskopie, sowie das Zusammenstellen einer Arbeitsdiskette mit Hilfsprogrammen, wie zum Beispiel einer RAM-Disk, bereiten so keine Probleme mehr.

Weiterhin fällt der Umfang der neuen Dokumentation auf, die nun die Bezeichnung Handbuch redlich verdient. Auf weit über dreihundert Seiten erklärt es nicht nur alle Aspekte von Editor, Compiler und Linker, sondern auch sämtliche Funktionen in den mitgelieferten Unterprogramm-Bibliotheken für den Linker.

Die bisher etwas umständliche Bedienung der alten Fassung von ST-Pascal lag am Fehlen einer einheitlichen Benutzerschnittstelle für die einzelnen Programmteile. Hier zahlte sich nun die Kooperation mit dem amerikanischen Systemsoftwarehaus »Optimized Systems Software« aus, das allen Besitzern eines 8-Bit-Ataris gut bekannt ist. Die kalifornische Firma steuerte eine Kommando-Shell bei, ähnlich der des Megamax-C-Compilers. Die Arbeitserleichterung beginnt schon damit, daß man komfortabel aus dem Pull-down-Menü »ST-Pascal plus« den Editor starten kann, nachdem man zuvor über den Dateiselektor den zu bearbeitenden Programmtext auswählt. Nach dem Start lädt der ST automatisch den Editor und den Source-Text. Der Editor ist nach wie vor eine reine TOS-Applikation. Es fehlen also nicht nur Pull-down-Menüs und Windows, sondern es verbietet sich auch die gleichzeitige Bearbeitung mehrerer Dateien. Hier setzt nach wie vor der Editor des GST-C-Compilers Maßstäbe.

Als schwerwiegenderer Mangel zählt allerdings die Begrenzung der maximalen Zeilenlänge auf 79 Zeichen. Das verhindert beispielsweise, den langen



Informative Compiler-Fehlermeldungen

```
File: A:\COMPARE.PAS                         free : 331K line:     2 insert
PROGRAM FileCompare;

CONST
 {$I GEMCONST.PAS}

TYPE
 {$I GEMTYPE.PAS}
 tftype = FILE of TEXT;

VAR
 pathname1,filename1,pathname2,filename2 : Path_Name;
 selec1,selec2,done : boolean;
 bt : integer;
 fn1,fn2 : tftype;
 b1,b2 : char;

{$I GEMSUBS.PAS}

BEGIN {Hauptprogramm}
 IF Init_Gem>=0 THEN
  BEGIN {Das eigentliche Programm}
   pathname1:='A:\*.*';
   pathname2:='A:\*.*';
   done:=false;
```

Der Editor: schnell, aber noch immer nicht unter GEM

Text für eine größere »Alert-Box« in einer Befehlszeile zu definieren. Ansonsten ist an dem Editor nichts auszusetzen. Er beinhaltet alle wichtigen Befehle, ist sehr schnell, beherrscht automatisches Einrücken von Programmzeilen und ist ideal auf das Pascal-System zugeschnitten. Das Drücken von F9 speichert die bearbeitete Datei und ruft den Compiler auf. Auf dem Bildschirm öffnet sich ein Fenster, in dem man die Arbeit des Compilers verfolgen kann. Es zeigt auch den Namen der bearbeiteten Datei. Tritt beim Compilieren ein Fehler auf, erscheint ein zweites Fenster, das die fehlerhafte Zeile anzeigt. Dem Anwender bieten sich nun zwei Alternativen: den Compiliervorgang fortzusetzen, um eventuelle weitere Fehler zu entdecken, oder ihn abzubrechen und gleich den fehlerhaften Programmtext in den Editor zu laden. Dabei steht der Cursor nach Laden der Datei genau auf der fehlerhaften Zeile. Damit ist der typische Arbeitsverlauf beim Schreiben eines Programmes voll automatisiert: Vom Verlassen des Editors bis zum erneuten Ändern des Programms braucht es nur einen einzigen Mausklick. So macht Programmieren Spaß.

Da der Compiler in dieser neuen Version nicht mehr auf das Einlegen der Original-Diskette wartet, kann man alle Dateien des Softwarepakets auch in eine RAM-Disk kopieren. In einer solchen Gerätekonfiguration – 1 MByte Speicherplatz wäre empfehlenswert, jedoch nicht unbedingt notwendig – spielt das Pascal-System erst seine ganzen Fähigkeiten aus. Zwischen dem Beenden der Programmeingabe und dem Korrigieren der aufgetretenen Fehler vergehen bei kürzeren Programmen keine 20 Sekunden. Das kommt trotz des aufwendigeren Konzepts mit separatem Linkvorgang schon fast an die Geschwindigkeit von Turbo-Pascal auf dem IBM-PC heran.

Obwohl Compiler als auch Linker in die GEM-Umgebung eingebunden sind, sind sie auch nach wie vor als separate TOS-Applikationen lauffähig. Beim Compiler handelt es sich um die Version 1.4[5a] des bekannten ST-Pascal-Compilers, und auch der Linker ist ein alter Bekannter: der bewährte »Fastlink«. Beide Programmteile weisen gegenüber der letzten Version von »ST Pascal« nur geringfügige Änderungen auf, so daß die Leistungen beim Compilieren, Linken und in der Ablaufgeschwindigkeit nicht nennenswert von Fastlink abweichen.

Die Programmierung von GEM-Applikationen war in ST-Pascal aus zweierlei Gründen kein einfaches Unterfangen. Einerseits mußte man alle GEM- und AES-Prozeduren zunächst als externe C-Prozeduren deklarieren und hatte dann andererseits noch immer das Problem der fehlenden Dokumentation der Unterprogramme. Bei »ST-Pascal plus« begnügte man sich nicht damit, nur die fehlenden Informationen nachzuliefern, sondern überarbeitete auch kurzerhand alle GEM-Aufrufe und packte sie in neue, einfacher zu benutzende Kommandos. Dazu gehören die Standard-Graphikbefehle des VDI und alle Funktionen des AES. Als Beispiele seien hier »BRING_TO_FRONT« (Fenster aktivieren), »DRAW-STRING« (Text an relativer Position im Fenster ausgeben) oder »Get_In_File« (Dateiselektor aufrufen) genannt. Über spezielle Routinen, »Aes_Call« und »Vdi_Call«, kann man im Bedarfsfall alle weiteren GEM-Routinen aufrufen und damit sehr tief in das Betriebssystem einsteigen.

Ein interessanter Aspekt der erweiterten GEM-Bibliotheken ist die nun wesentlich einfachere Erzeugung von Ressource-Objekten durch das Programm. So kommt man unter Umständen auch ohne ein Ressource-Construction-Programm aus, das nach wie vor im Programmpaket fehlt. Auch die von der Vorgängerversion her bekannten Befehle für die schnelle Line-A-Grafik fielen nicht unter den Tisch. Die auf der Diskette mitgelieferten Programmbeispiele legen ein deutliches Zeugnis dafür ab, um wieviel leichter die Programmierung mit den neuen Unterprogramm-Bibliotheken fällt. Damit kann sich der Programmierer wieder mehr auf das Wesentliche konzentrieren und die Handhabung der GEM-Befehle getrost dem Pascal-Compiler überlassen. Dennoch ist der direkte Zugriff auf die GEM-Routinen nicht verbaut.

»ST Pascal plus« eignet sich nicht nur für die Entwicklung größerer GEM-Programme, sondern bietet sich wegen der einfachen Bedienung und der guten Dokumentation auch für den Pascal-Einsteiger an, zumal die Kundenbetreuung für den Hersteller CCD nicht nur ein Gerede ist. Wer nicht unter GEM programmieren will, sollte auch das alte ST-Pascal-System, das ab sofort nur noch 148 Mark kostet, in Erwägung ziehen.

Mit »ST Pascal plus« setzen CCD und OSS den momentanen Standard für Pascal-Compiler auf dem ST. Nun ist Borland gefordert, mit der lange erwarteten Version von Turbo-Pascal für die Computer mit dem Mikroprozessor 68000 zu zeigen, wer bei den ST-Programmiersprachen die Nase vorne hat. Sie müssen sich aber beeilen, denn eine Programmiersprache der neuen Generation, »Personal-Prolog«, hat OSS für den amerikanischen Markt bereits angekündigt...

(Julian Reschke/hb)

Info: CCD, Schöne Aussicht 41, 6229 Walluf, Tel.: 06123/73881

Fortran auf QL und ST

Fortran ist die Sprache der Wissenschaft. Altbewährt, besticht sie doch durch außergewöhnliche Eigenschaften. Das Fortran von Prospero haben wir auf Herz und Nieren getestet.

Die preiswerteste Kombination, mit Fortran auf dem eigenen Computer zu programmieren, stellt derzeit Pro Fortran-77 von Prospero auf dem Atari St dar. Derselbe Hersteller bietet auch für den QL eine Version an.

Bei der Leistungsfähigkeit müssen Sie keine Abstriche machen. Einerseits bietet der Fortran-Compiler softwareseitig alle Eigenschaften professioneller Implementationen, zum anderen glänzt der ST mit riesigem Speicher und einer Rechengeschwindigkeit, die kaum Wünsche offenläßt.

Das Prospero-Fortran wird auf einer einseitigen Diskette ausgeliefert. Auf dieser Diskette befinden sich Compiler, Linker, eine residente Bibliothek sowie Konfigurations- und Beispielprogramme.

Wie der eifrige Programmierer sofort feststellt, fehlt ein Editor. Das ist aber nicht weiter tragisch. Quelltexte für Fortran zu schreiben gestaltet sich mit fast jedem Textverarbeitungsprogramm recht komfortabel. Als Editoren für Fortran eignen sich zum Beispiel Protext, 1st Word oder der GST-Editor. Auch finden Sie auf der Service-Diskette zu dieser Ausgabe einen hervorragend geeigneten Editor. Im folgenden beschreiben wir kurz die Funktion der einzelnen Programme, die Sie auf der Pro-Fortran-Diskette erhalten.

Die Beispielprogramme zeigen, daß über die Programmierung in Fortran hinaus auch die Anwendung von GEM unterstützt wird. Hierauf kommen wir später bei der Besprechung des Compilers zurück.

Damit ein Programm beim Compilieren und Linken keinen unnötig langen Code erhält, wird zu Beginn der Arbeit mit Pro Fortran eine residente Bibliothek mit Standardprozeduren geladen. Es empfiehlt sich, das Programm PRL.PRG in einen Auto-Ordner zu kopieren, so daß es bei der Initialisierung des Computers gleich gestartet wird.

Die Programme FCONFIG.PRG legen die Laufwerke fest, in denen die einzelnen Programme zum Compilieren und Linken zu finden sein müssen. Außerdem werden hiermit die Voreinstellungen des Compilers neu initialisiert. Beim Compilieren erhält das Programm am Ende eine Warteschleife, die es erst auf Tastendruck verläßt und zum GEM-Desktop zurückkehrt. Das Programm RCONFIG.PRG verändert ein Programm so, daß es diese Warteschleife nicht mehr durchläuft. Mit dem Programm PROLIB.PRG lassen sich compilierte Unterprogramme zu einer Bibliothek zusammenfassen. Ein großer Vorteil dieser Fortran-Version ist der geringe Speicherbedarf. Editor, Compiler und Linker finden in einer RAM-Disk Platz, was einem großen Geschwindigkeitsvorteil bringt. Compilieren und Linken von einer RAM-Disk aus liegen im Sekundenbereich, während die gleichen Operationen von der Diskettenstation mit mehreren Minuten zu Buche schlagen. Hier zeigt sich auch einmal mehr der Vorteil ungeschützter Programme: Pro Fortran-77 läßt sich problemlos in die RAM-Disk kopieren. Die wichtigste Komponente einer Hochsprache ist der Compiler. Er gibt den Ausschlag für die Leistung des gesamten Systems. Deshalb wollen wir ihn im folgenden genauer unter die Lupe nehmen:

Compiler mit vollem Standard

Pro Fortran-77 besitzt einen Zwei-Pass-Compiler. Das heißt, daß der Quelltext zunächst in einen Zwischencode übersetzt wird und anschließend in den Assembler-Objektcode.

Der Compiler verarbeitet den vollen Sprachumfang nach dem Standard, den die ANSI (Amerikanisches Institut für Normung und Standardisierung) als X3.9-1978 einführte. Pro Fortran enthält somit kein Subset, sondern den vollen Fortran-77-Sprachumfang. Zusätzlich zu den herkömmlichen Befehlen ist außerdem der gesamte GEM-Sprachumfang mit sämtlichen VDI- und AES-Aufrufen implementiert. Damit erzeugen Sie problemlos umfangreiche Grafiken in Fortran oder benutzen Fenstertechnik und Maussteuerung. Durch Einsatz dieser Befehle bleibt die Kompatibilität zu anderen Fortran-Computern jedoch auf der Strecke.

Der Compiler generiert 68000-Negative-Code. Das heißt, die Programme müßten normalerweise ohne eine spezielle Programmumgebung laufen. Daß dem nicht so ist, liegt an der residenten Bibliothek, die vor Arbeitsbeginn in den Hauptspeicher geladen werden muß. Wollen Sie beispielsweise Programme weitergeben, so ist diesen die residente Bibliothek (PRL.PRG) hinzuzufügen.

Der Compiler benötigt für seine Arbeit einen freien Speicherplatz von nur 96 KByte. Er läuft also auch auf einem 260ST ohne Probleme. Der »kleine« Atari erlaubt das Arbeiten mit der RAM-Disk hingegen nicht. Der Compiler ist aber auch mit einer einseitigen Disketten-Station lauffähig, so daß schon eine Minimalausstattung an Hardware ausreicht. Es handelt sich bei dem Compiler um ein TOS-Programm, Sie müssen also alle Befehle per Tastatur eingeben. Das Arbeiten selbst erfordert mindestens die zwei Programme PROFOR1.PRG und PROFOR2.PRG, allerdings empfiehlt es sich, das Steuerprogramm F77.PRG zum Compilieren zu benutzen. Es steuert dann selbständig beide Durchgänge. Außerdem wird eventuell (und das heißt hier in den meisten aller Fälle) das File PROFOR.ERR benötigt, das für die Ausgabe der Fehlermeldungen verantwortlich zeichnet. Zum Compilieren starten Sie also F77.PRG (es zahlt sich aus, die Extension dieses Programms in ».TOS« umzubenennen, da Sie so einen Cursor erhalten). Nach Anzeige des Copyrights werden Sie aufgefordert, den Namen des Workfiles (das ist das Programm, das Sie compilieren wollen) einzugeben. Das Workfile muß auf ».FOR« enden. Nach der Eingabe erscheint die folgende Zeile:

»G-console output to LOG file ? (Y/N/.)«.

Diese Meldung fordert Sie auf, den ersten Compiler-Befehl zu benennen. Antworten Sie mit »Y«, so wird er aktiviert. Geben Sie ».« ein, so arbeitet das Programm mit festgelegten Voreinstellungen (Defaults). In diesem Fall wird keine der zusätzlichen Funktionen aktiviert, Sie erreichen also dasselbe, wenn Sie bei jeder Frage mit »N« antworten. Nun zu den wahlweisen Einstellungen im einzelnen:

G-console output to LOG file:

Dieser Befehl bewirkt, daß sämtliche Meldungen des Compilers nicht nur auf dem Bildschirm ausgegeben, sondern auch in ein File mit dem Namen Workfilename.LOG auf Diskette gespeichert werden. Es empfiehlt sich, diese Ein-

stellung bei vielen Fehlern im Programm vorzunehmen.

I-range checks on subscripts:
Dieser Befehl bindet in das compilierte Programm eine zusätzliche Routine ein. Sie gibt eine definierte Fehlermeldung aus, sobald ein zu niedrig dimensioniertes Variablenfeld überläuft. Diese Einstellung ist sinnvoll, wenn sich ein Programm in der Entstehungsphase befindet und man diese Art Laufzeitfehler vermeiden will. Sie bewirkt allerdings einen längeren Programm-Code sowie eine längere Ausführungszeit des Programms.

A-range checks on assignments:
Mit diesem Befehl stellen Sie sicher, daß der Bereich der Werte, die den INTEGER-Variablen zugewiesen werden, nicht überschritten wird. Hierbei wird der erzeugte Code ebenfalls länger.

N-track source names & line numbers at run time:
Dies ist die TRACE-Funktion des Compilers. Tritt irgendwo im Programm ein Laufzeitfehler auf, so veranlaßt dieser Befehl den Computer, die Zeilennummer auszugeben, in der der Fehler aufgetreten ist.

M-map:
Mit dieser Funktion wird eine Referenzliste für alle im Programm vorkommenden Namen, das heißt Variablen, Prozeduren und Funktionen erzeugt, die dann auf Diskette mit dem Kürzel ».MAP« gespeichert wird. Hiermit überprüfen Sie, ob in dem Programm Typenkonflikte bestehen oder ob irgendwelche Variablen ohne Funktionen vorhanden sind.

L-source listing:
Hiermit wird während des Compilierens ein Listing des Quelltextes erzeugt. Dabei erhält jede Zeile eine Zeilennummer sowie die relative hexadezimale Adresse, die den Beginn dieser Zeile im Object-Code anzeigt. Das Listing wird mit dem Kürzel ».PRN« auf Diskette abgelegt.

U-report undeclared Variables:
In Fortran ist eine implizite Variablendeklaration vorgesehen, das heißt, alle Variablen mit den Anfangsbuchstaben I bis M, sofern nicht anders vereinbart, werden als Integer erkannt. Alle anderen Variablen werden als Real-Typen verarbeitet. Machen Sie von den Impliziten im Programm Gebrauch, so kann es vorkommen, daß eine falsch geschriebene Variable den Compiler nicht abbrechen läßt. Eine schwierige Fehlersuche ist meist die lästige Folge. Dieser Befehl veranlaßt den Compiler, alle nicht deklarierten Variablen auszugeben.

T-INTEGER means INTEGER*2:
Diese Funktion wahrt die Kompatibilität zu Computern mit anderen Prozessoren. Der 68000 erkennt den Integer-Typ normalerweise als Integer*4, das heißt als 4-Byte-Integer. Beim Wählen dieses Befehls werden alle nicht explizit als INTEGER*4 definierten Integer-Variablen als INTEGER*2, also als 2-Byte-Integer, aufgefaßt.

C-compact object code:
Dieser letzte Befehl veranlaßt den Compiler, einen (noch) kürzeren Code zu erzeugen. Er ist eigentlich überflüssig, da der Compiler ohnehin bereits einen sehr kurzen Code liefert und dieser Befehl den Code nur um wenige Bytes, etwa ein Prozent der Programmlänge, verkürzt. Allerdings wird die Ausführungsgeschwindigkeit merklich vermindert.

Neben der schon erwähnten Möglichkeit auf VDI und AES zurückzugreifen, unterstützt der Compiler noch weitere Funktionen, die vom Fortran-77-Standard abweichen. So bietet er den Zugriff auf GEMDOS-, BIOS- und XBIOS-Funktionen. Es ist allerdings recht umständlich, diese Funktionen in einem Programm zu nutzen.

Die RANDOM-Funktion erzeugt eine Pseudo-Zufallszahl zwischen Null und eins.

Mit dem Befehl IADDR (Variable) ermitteln Sie die Adresse einer Variablen. Als Ergebnis wird eine 32-Bit-Zahl ausgegeben.

»IPEEK (iaddress)« ist eine aus BASIC bekannte Funktion. Sie ergibt den Wert des Bytes in der Speicherzelle iaddress.

Die Funktion »POKE (iaddress, ival)« ist gleichfalls ein Äquivalent zu der POKE-Funktion in BASIC. Hier wird ein Byte »ival« an die Stelle iaddress geschrieben.

Der Befehl EXECPG startet aus einem Programm heraus ein anderes Programm. Sie verlassen das aufgerufene Programm wieder mit dem Befehl EXITPG. Sehr nützlich ist auch der Befehl AFFIRM. Es erscheint eine Meldung oder eine Frage auf dem Bildschirm, die mit »Y« oder »N« zu beantworten ist. Je nach Antwort ist das Ergebnis dieser Funktion True oder False.

Mit dem Befehl »TIME« (h, min, sec, hundertstel) schließlich fragen Sie die interne Uhr Ihres Computers ab.

Sobald Sie auf diese zusätzlichen Befehle zugreifen, sollten Sie sich im klaren darüber sein, daß Ihre Programme nicht mehr zum Fortran-77-Standard kompatibel sind.

Geschwindigkeit ist keine Hexerei

Es hat sich eingebürgert, für Geschwindigkeitstests von Programmiersprachen sogenannte Benchmarks zu verwenden. Das sind Programme, die nur darauf zugeschnitten sind, bestimmte spezifische Eigenschaften des Compilers zu nutzen. Unser Test wurde mit drei Fortran-Compilern auf dem IBM-XT mit einem 8087-Coprozessor durchgeführt. Fünf Benchmark-Programme prüften Rechengeschwindigkeit und -genauigkeit der Compiler. (Wie diese Benchmark funktionieren, ist ausführlich in unserer Schwesterzeitschrift Computer persönlich, Ausgabe 1/86, Seite 92 bis 96, zu lesen.)

BENCH1 und BENCHF testeten die Rechengeschwindigkeit der Fließkomma-Arithmetik, BENCHA hingegen die Ausführungszeit von häufig vorkommenden Programmteilen (Schleifen, Zuweisungen). Mit BENCH2 und BENCH4 schließlich wird die Rechengenauigkeit überprüft.

Ryan Mcfarland-Fortran (IBM-XT) (bestes getestetes Fortran auf dem IBM).		
BENCH1:	Ausführungszeit:	8 Sek.
	Programmlänge:	29309 Byte
BENCH2:	Programmlänge:	28784 Byte
BENCH4:	Programmlänge:	28665 Byte
BENCHA:	Ausführungszeit:	21 Sek.
	Programmlänge:	170988 Byte
BENCHF:	Ausführungszeit:	115 Sek.
	Programmlänge:	10808 Byte
Pro-Fortran (Atari-ST):		
BENCH1:	Ausführungszeit:	110 Sek.
	Programmlänge:	8124 Byte
BENCH2:	Programmlänge:	6598 Byte
BENCH4:	Programmlänge	6108 Byte
BENCHA:	Ausführungszeit:	8 Sek.
	Programmlänge:	4560 Byte
BENCHF:	Ausführungszeit:	378 Sek.
	Programmlänge:	4642 Byte

Wie Sie aus obenstehender Tabelle erkennen, ist der Pro-Fortran-Compiler dem Fortran auf dem IBM-XT in der Codelänge und der reinen Ausführungsgeschwindigkeit (in BenchA etwa um den Faktor 2,6) überlegen. Die Codelänge ist zirka 4mal kürzer als auf dem IBM-XT. Dies liegt an der residenten Library des Pro-Fortran-Compilers. Da diese ständig im Speicher des ST zur Verfügung steht, muß weniger zum Programmcode »hinzugelinkt« werden. Die Rechengenauigkeit liegt in demselben Rahmen wie die des Ryan-McFarland-Fortran.

Allerdings läßt die Rechengeschwindigkeit der Fließkomma-Arithmetik



Bücher zu Programmiersprachen

P. Lüke
Turbo-Pascal
290 Seiten

Pascal besitzt neben den von Standard-Pascal her bekannten Sprachelementen eine Reihe von Ausstattungsmerkmalen, die die Verwendung eines Compilers auf einem bisher bekannten Komfortniveau erlauben. Dieses Buch enthält eine logisch aufgebaute Einführung in die Benutzung und Anwendung von Turbo-Pascal (einschließlich Version 3.0): Programmstruktur · Syntaxdiagramme · anwenderdefinierte Funktionen und Prozeduren · Turbo-Toolbox · Turbo-Lader · die Implementation von MS-DOS, CP/M-86, CP/M · Arithmetik · Grafik, Farbe, Sound und Window-Technik.
didaktisch hervorragende Aufbau dieses Buches ermöglicht auch dem Anfänger ohne Vorkenntnisse, die Vorteile von Turbo-Pascal schnell und effektiv zu nutzen.
Best.-Nr. MT 90150
ISBN 3-89090-150-6
-/sFr. 45,10/öS 382,20





J. Purdum/T. Leslie/A. Stegemöller
Die C-Programmbibliothek
Februar 1986, 361 Seiten

Dieses Buch erspart dem C-Programmierer Stunden mühseliger Kleinarbeit und hilft, effizientere Programme zu schreiben.
Der erste Teil zeigt, wie man zu universellen Bibliotheksfunktionen kommt und gibt Tips, wie C noch wirkungsvoller eingesetzt werden kann. Der zweite Teil enthält eine Reihe ausführlich erklärter C-Funktionen als wertvolle Ergänzung Ihrer Programmbibliothek. Dazu gehören unter anderem ein Terminalinstallationsprogramm, mehrere Sortier-Algorithmen und ein Satz ISAM-Funktionen. Um die Anwendung der Funktionen zu verdeutlichen, enthält das Buch einige Programmbeispiele.
- Die Programmbibliothek wendet sich an Leser mit Grundkenntnissen von C. Die gezeigten Programme und Funktionen sind so gehalten, daß sie Rechner- und Compiler-unabhängig eingesetzt werden können.

Best.-Nr. MT 90133
ISBN 3-89090-133-6
DM 69,-/sFr. 63,50/öS 538,20













H. Schwanke
Programmieren
Microsoft COBOL
1986,
Seiten

Schwerpunkt dieses Buches liegt in der kommerziellen Anwendung der Datenverarbeitung. An zahlreichen Beispielprogrammen werden COBOL-Sprachelemente und Techniken für Dialogprogrammierung die Verarbeitung großer Datenmengen gezeigt.
Best.-Nr. MT 90108
ISBN 3-89090-108-5
-/sFr. 71,80/öS 608,40

COBOL
Handbuch für
Mikrocomputer
überarbeitete Auflage
297 Seiten
Best.-Nr. MT 90162
ISBN 3-89090-162-X
-/sFr. 47,80/öS 405,60

Prof. Nestle/E. Ostertag
Kleiner Sprachführer
BASIC - LOGO - PASCAL
April 1986, 199 Seiten

Der Einstieg in eine Sprache ist mühsam, ebenso das Umsteigen von einer Sprache auf die andere. Allein mit den Handbüchern ist das oft nicht zu bewältigen. An diesem Punkt setzt das Buch mit seinen Beispielen an: Zu einem gegebenen Bildschirmaufbau bringt das Buch Problemlösungen in jeder der drei Sprachen BASIC, LOGO und PASCAL. Der Leser, der eine der Sprachen beherrscht, kann deshalb zunächst einmal die Programmierung in der ihm bekannten Sprache nachvollziehen. Dabei erkennt er die Teilaufgaben, die gelöst werden müssen, und kann dann Schritt für Schritt deren Lösung in der anderen Sprache verfolgen.
Best.-Nr. MT 90160
ISBN 3-89090-160-3
DM 39,-/sFr. 35,90/öS 304,20

N. Graham
Programmieren mit dem IBM-PC: BASIC
1984, 442 Seiten

Früher oder später wird wohl in jedem PC-Besitzer oder PC-Benutzer der Wunsch erwachen, selbst Programme zu schreiben und so die Maschinen besser zu nutzen. Die Wahl der Programmiersprache liegt nahe: denn der PC verfügt über ein leistungsfähiges BASIC. Dessen Beherrschung wird Ihnen in diesem Buch Schritt für Schritt nahegebracht. Auch die Dateiverarbeitung, welche gerade für Anfänger voller Fallstricke ist, wird hier eingehend besprochen. Sie ist Voraussetzung für viele kommerziell interessante Programme. Ein Kapitel ist den Techniken des Programmentwurfs gewidmet.
Best.-Nr. MT 663
ISBN 3-922120-97-0
DM 58,-/sFr. 53,40/öS 452,40

P. M. Chirlian
Der Einstieg in C
1985, 290 Seiten

C ist der neue Star unter den Programmiersprachen. Zum Erlernen dieser vielseitigen und mächtigen Programmiersprache, die gerade ihren Siegeszug durch Europa antritt, finden Sie in diesem Buch alles, was Sie wissen müssen! Gleich von Anfang an schreibt der Leser seine eigenen Programme.
Besonderes Augenmerk legt der Autor auf die wichtige Methode des strukturierten Programmierens, für das sich C bestens eignet. Auch das wichtige Thema Datentypen kommt nicht zu kurz. Die im Anhang aufgeführte Syntaxübersicht macht das Buch zu einem wertvollen Hilfsmittel bei der praktischen Arbeit.
Best.-Nr. MT 90086
ISBN 3-89090-086-0
DM 60,-/sFr. 55,20/öS 468,-

L. J. Scanlon
Die Assemblersprache des IBM-PC & XT
1985, 351 Seiten

Trotz der Verfügbarkeit vieler höherer Programmiersprachen: Wer wirklich über seinen Computer Bescheid wissen, wer sich vielleicht sogar in die Höhen der Systemprogrammierung (z.B. Disketteneditor) vorwagen will, der kommt auch auf so komfortablen Maschinen wie den Personal Computern von IBM um Maschinensprache nicht herum.
Best.-Nr. MT 654
ISBN 3-922120-88-1
DM 74,-/sFr. 68,10/öS 577,20

D. A. Lien
BASIC-Programmierung PC-10/PC-20
1985, 488 Seiten

In locker-entkrampfter Sprache – nach bester amerikanischer Lehrbuch-Manier, mit didaktisch - systematischem Aufbau – der Autor verfügt über jahrelange Lehrerfahrung, und mit vielen praktischen und interessanten Beispielen – selbstverständlich mit Lösungen, führt D. A. Lien hier in die Programmierung mit BASIC ein.
Best.-Nr. PW 80366
ISBN 3-921803-66-7
DM 59,-/sFr. 54,30/öS 460,20



Fragen Sie Ihren Buchhändler nach unserem kostenlosen Gesamtverzeichnis mit über 200 aktuellen Computerbüchern und Softwareprogrammen. Oder fordern Sie es direkt beim Verlag an!

Markt & Technik-Fachbücher erhalten Sie bei Ihrem Buchhändler

Bestellungen im Ausland bitte an den Buchhandel oder an untenstehende Adressen.
Schweiz: Markt & Technik Vertriebs AG, strasse 3, CH-6300 Zug, ☎ 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, A-1091 Wien, ☎ 0222/48 15 38-0

Irrtümer und Änderungen vorbehalten.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

noch einige Wünsche offen. So zeigte sich der IBM-XT bei doppeltgenauen Rechnungen zirka um den Faktor 13 schneller, bei der einfachgenauen Arithmetik um den Faktor 3. Dies verwundert auch nicht weiter, denn schließlich besitzt der IBM einen Arithmetik-Coprozessor, während der 68000 alle anfallenden Aufgaben im Alleingang erledigt. Unter diesem Augenmerk schneidet der Atari sogar hervorragend ab. Spätestens bei Einführung des 68881, dem arithmetischen Coprozessor der 68000er-Familie, wird der Atari dann die Nase wieder vorn haben.

Vergleicht man allerdings die Anschaffungspreise eines Fortran-Systems auf IBM-Basis mit denen eines Atari ST, so ergibt sich hier eine Einsparung von rund 75 Prozent zugunsten des Atari. Etwa 8000 Mark für die Hardware und zirka 2000 Mark für die Software müssen Sie beim IBM-XT berappen. Dagegen stehen zirka 2000 Mark für die Hardware und 490 Mark für Pro Fortran-77 beim ST. Die Fortran-Version für den QL schlägt sogar mit nur 330 Mark zu Buche.

Als Linker erhalten Sie den bewährten GST-Linker. Dieser bindet Fortran-77-Programme mit Assembler- oder C-Routinen problemlos zusammen.

Professionell und preiswert

Pro Fortran-77 bietet Leistungen, die professionellen Systemen in keiner Weise nachstehen. Das Programm wartet mit einer ganzen Reihe von gut aufeinander abgestimmten Programmen, einem erstklassigen Compiler und einem bewährten Linker auf. Der Preis von 490 Mark für die Atari-Version und 330 Mark für die QL-Version ist somit als preisgünstig einzustufen.

(Christian Träger/
Michael Zwenger/Matthias Rosin/hb)

```
BENCH1:

      REAL*8 Z,A,B,Y
      INTEGER*2 I,ILOOP
      INTEGER*4 DUMMY,MINI,MINO,SECI,SECO
      CALL TIME(DUMMY,MINI,SECI,DUMMY)
      ILOOP=5000
      Z=0
      DO 1 I=1, ILOOP
        A=I
        B=DTAN(DATAN(DEXP(DLOG(DSQRT(A*A)))))/A-1
        Z=Z+B*B
    1 CONTINUE
      Y=DSQRT(Z/ILOOP)
      CALL TIME(DUMMY,MINO,SECO,DUMMY)
      WRITE(*,*) Y
      WRITE(*,*) MINI,SECI
      WRITE(*,*) MINO,SECO
      STOP
      END

BENCH2:

      REAL*8 A,E,N
      INTEGER*2 I
      N=1
      A=1
      DO 10 I=1,16
      E=(A+A/N)**N
      WRITE(*,90) N,E
   90 FORMAT(1H ,'N= ',F18.1,5X,'E= ',F18.16)
      N=N*10
   10 CONTINUE
      STOP
      END

BENCH4:

      INTEGER*4 N
      REAL*8 X,Y
      N=1
   10 X=1.0/DBLE(N)
      Y=(DEXP(-X)-1.0+X)/(X*X)
      WRITE(*,20) X,Y
   20 FORMAT(1H ,F10.8,5X,F18.16)
      IF (N.GT.10000000) GOTO 30
      N=N*10
      GOTO 10
   30 CONTINUE
      STOP
      END

BENCHA:

      PROGRAM TEST
      COMMON /GROSS/FELD(40000)
      INTEGER*4 I,DUMMY,MINI,MINO,SECI,SECO
      CALL TIME(DUMMY,MINI,SECI,DUMMY)
      DO 1 I=1,40000
    1 FELD(I)=1.0
      DO 2 I=1,40000
      FELD(I)=FELD(I)*FELD(40000-I)
    2 CONTINUE
      CALL TIME(DUMMY,MINO,SECO,DUMMY)
      WRITE(*,*) MINI,SECI
      WRITE(*,*) MINO,SECO
      STOP
      END

BENCHF:

      PROGRAM FLT
      REAL*8 X,Y,Z
      INTEGER*4 I,J,D,MI,MO,SI,SO
      CALL TIME(D,MI,SI,D)
      DO 1 I=1,256000
      J=256000-I
      X=FLOAT(I)
      Y=FLOAT(J)
      Z=Y/X
      X=Y-Z
      Y=Z*X
      Z=Y+X
    1 CONTINUE
      X=Z+Y
      CALL TIME(D,MO,SO,D)
      WRITE(*,*) MI,SI
      WRITE(*,*) MO,SO
      STOP
      END
```

Die Benchmarks lassen den Atari ST gut abschneiden

Aufbruch zu Modula-2

TDI-Modula auf dem ST ist inzwischen ein fester Begriff. Stärken und Schwächen der neuen Version, die nun auch GEM unterstützt, offenbart unser Testbericht.

Wer moderne Computer benutzt, will auch moderne Sprachen verwenden. So erscheint es ganz natürlich, daß auch unter den ST-Freaks die modulare Sucht grassiert. Die kürzlich erfolgte Preissenkung auf 350 Mark und die neue symbolische Benutzeroberfläche von TDI-Modula, sollten den Einstieg zur Verlockung machen.

Glanz mit kleinen Kratzern

Programmiersprachen werden oft nach der Güte ihrer Oberflächen und der verfügbaren Programmierwerkzeuge beurteilt. Die Freude an einer vorbildlich strukturierten und modularen Sprache wird durch langweilige Compiler, umständliche Handhabung und mangelhafte Werkzeuge (Editor, Debugger) verdorben. Hier glänzt TDI-Modula auf den ersten Blick. Wenn Sie das Programm M2DESK.PRG starten, erscheint eine Schreibtischoberfläche mit recht eigenwilligen Pictogrammen (Bild 1), die den Editor, Compiler, Linker und Debugger darstellen. Sie starten diese Programme durch Anklicken der Pictogramme und Eingabe der Dateinamen. Es geht aber noch eleganter. Zum Lieferumfang gehört auch ein Accessory »Modula-2 Options«.Hiermit lassen sich verschiedene Systemparameter einstellen (Bild 2 und 3). Besonders wichtig für die Arbeit sind die vier voreinstellbaren Pfadnamen im Bild 3, die nach Betätigen des Schalters »System« erscheinen. Der erste Pfad führt in dasjenige Directory, das auf dem Schreibtisch angezeigt wird.

Die Anzeige, die in Bild 1 zu erkennen ist, wurde sehr praxisgerecht gestaltet. Es werden nur die zur Arbeit mit dem Programmentwicklungssystem gehörenden Dateien angezeigt, das sind die mit den Namens-Erweiterungen »DEF«, »MOD«, »SYM« und »PRG«. Die Quellen der Definitions- und Implementierungsmodule werden als Dokumente dargestellt, die SYM-Files (compilierte Schnittstellenbeschreibung) als eine Art Kristallsymbol. Alle zu einem Programm gehörenden Symbole mit dem gleichen Namenshauptteil erscheinen in einer Reihe nebeneinander. Rechts in Bild 1 sind die compilierten Link-Files als Puzzlestücke und die gelinkten PRG-Files als fertig zusammengesetztes Puzzle dargestellt.

Um ein Symbol zu bearbeiten, klicken Sie es an. Das führt bei den DEF- oder SYM-Pictogrammen dazu, daß der Editor aufgerufen wird, der automatisch diese Textdatei lädt. Klicken Sie auf das SYM-Zeichen, so startet der Compiler und übersetzt automatisch das zugehörige DEF-File neu. Auch wenn der Platz eines Symbols leer ist, führt das Anklicken dieser Position dazu, daß es – wenn möglich – durch Aufruf von Compiler oder Linker erzeugt wird. Wenn das LNK- und PRG-Symbol fehlen oder ungültig sind, werden sogar mit einem einzigen Maustastendruck nacheinander Compiler und Linker gestartet, um das fertig gelinkte Programm zu erzeugen.

Wenn durch das Neubearbeiten eines linksstehenden Symbols die weiter rechts angezeigten ungültig geworden sind, erscheinen sie schattiert (erkennbar am LNK-File zum Modul Drucker in Bild 1). Das passiert zum Beispiel nach dem Editieren eines DEF- oder MOD-Files, denn nun muß ja das LNK-File neu compiliert und eventuell auch neu gelinkt werden. Hier gibt es leider eine Einschränkung: wenn Sie die Shell (also den Modula-Schreibtisch) vorübergehend verlassen, um ins TOS zurückzukehren, hat das System anschließend vergessen, welche Dateien ungültig geworden sind. Weitere Schwächen sind, daß das Starten von Programmen (über das Pull-down-Menü File) nicht gelingt, wenn sich die Programme in einem zweifach geschachtelten Ordner befinden. Ferner wird bedauerlicherweise nach einem Compilerlauf, bei dem ein Fehler aufgetreten ist, anschließend der Linker gestartet.

Eigenwillig, aber verbessert

Der Editor arbeitet zwar mit einem (feststehenden) GEM-Fenster und Mausbedienung, ist aber ansonsten langsam, umständlich und wartet mit einigen seltsamen Eigenheiten auf. Die am lästigsten erscheinende ist, daß sich der Cursor nicht direkt in Zeilen steuern läßt, deren Ende links vom augenblicklichen Cursor-Standort liegt, sprich, die zu kurz sind. Sie müssen erst nach links fahren, bis der Cursor senkrecht auf die gewünschte Zeile abspringen kann. Andernfalls springt der Cursor zur nächsten genügend langen Zeile, die eventuell auf einer anderen Seite liegt, so daß die Textstelle, die man bearbeiten will, vom Bildschirm verschwindet.

Wichtig ist aber, daß der Editor inzwischen immerhin zuverlässig arbeitet. Die früher gelegentlich auftretenden Abstürze wurden nicht mehr beobachtet. Auch Suchen und Austauschen funktionieren inzwischen problemlos, der Umfang an Funktionen entspricht professionellen Ansprüchen.

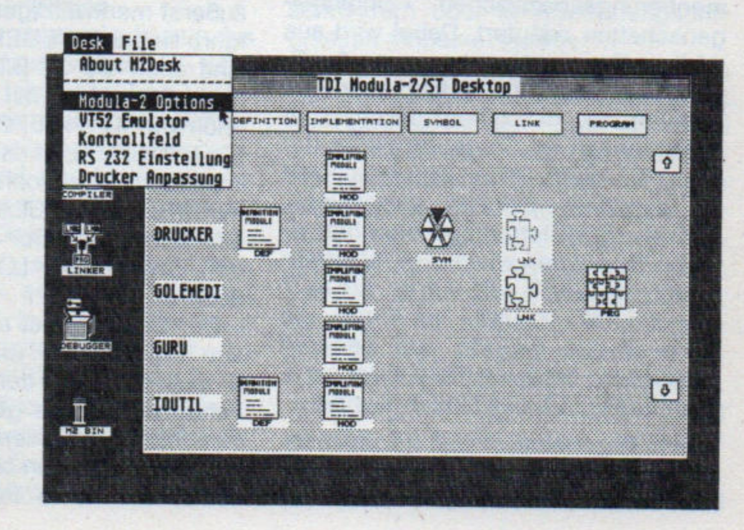


Bild 1. Das ist TDI-Modula. Die grafische Benutzeroberfläche wird vorbildlich genutzt.

Wie aus den Autoren-Vermerken hervorgeht, ist TDI-Modula im wesentlichen das Werk eines Zweierteams (normalerweise ein eher gutes Zeichen), nur ist das Schreiben eines flotten und bequemen Editors nicht gerade eine Stärke des Programmierer-Duos. Das erstaunt um so mehr, als es uns innerhalb von wenigen Wochen gelungen ist, in TDI-Modula einen sehr brauchbaren und schnellen Programmeditor zu schreiben, der in Bedienung und Funktion weitgehend dem von Turbo-Pascal entspricht (grafische Windows braucht man dazu nicht unbedingt). Der TLGE (The Little Golem Editor) editiert zwei Dateien gleichzeitig, zwischen denen Sie mit zwei Tastendrücken umschalten. Ich programmiere mittlerweile ausschließlich mit dem TLGE, lediglich für die Fehlersuche muß der TDI-Editor noch herhalten. Den TLGE erhalten Sie mit der Service-Diskette zu dieser Ausgabe.

Der TLGE ist der erste Baustein zu einem Programmier-Projekt »Golem«, das im nächsten 68000er-Sonderheft starten wird. Soviel sei hier bereits verraten: Es wird sich um eine Programmiersprache mit Zukunft drehen und jeder kann mitmachen. Aber kehren wir zurück.

Das sauber gedruckte Handbuch zum TDI-Modula enthält auf mehr als 350 Seiten eine Fülle von Informationen. Leider ist es nicht sonderlich übersichtlich. So finden Sie in den sehr zahlreichen Beschreibungen beziehungsweise Auflistungen der Bibliotheks-Modulschnittstellen nur schwer die gesuchte. Zwar gibt es ein Register mit Standardprozeduren und Namen aus den Bibliotheken, jedoch fehlen die entsprechenden Seitenverweise.

In dem ziemlich kurzen Abschnitt über den Compiler werden die implementierungsspezifischen Spracheigenschaften erläutert. Dabei wird aus Wirths Revisionspapier vom 1. 2. 84/14. 5. 84 zitiert, ich fand aber eine Erweiterung, die offensichtlich nicht implementiert ist. So gestattete es der Compiler nicht, einer Stringvariablen ein einzelnes Zeichen zuzuweisen. Andere Erweiterungen beziehungsweise Revisionen wie der »überzählige Balken« in CASE, die Kürzel für NOT (~) und AND (&) und – vor allem – die Revision der Exporte aus getrennt übersetzten Modulen (Fortfall der expliziten EXPORT-Liste) sind vorhanden.

Sehr angenehm ist, daß »große Mengen« (das heißt solche mit mehr als 32 Elementen) implementiert sind, TDI erlaubt bis zu 65536 Elemente in einer Menge. Damit ist auch SET OF CHAR fast uneingeschränkt deklarierbar, was die Programmierung von Menüabfragen oder kontrollierten Eingabefeldern sehr vereinfacht.

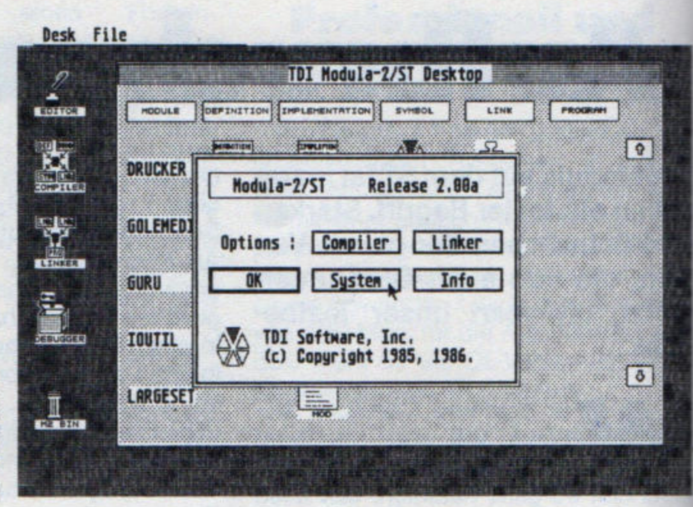


Bild 2. Das Menü zeigt die jeweiligen Symbole für Compiler, Linker und Filesystem

Daß in FOR- Schleifen nur Schrittgrößen bis 32767 und in CASE nicht mehr als 32767 Marken verwendet werden dürfen, wird in der Praxis nicht stören, unangenehm aufgefallen ist mir aber, daß der Compiler nicht mehr als 32 KByte statische globale Variable anlegen kann – das ist auf dem Atari ST wenig. Allerdings kann man über Zeigervariable auf dem Heap beliebig viele Daten verwalten, so legt beispielsweise der Editor TLGE einen 320 KByte großen Textpuffer an.

Die einschneidendste Beschränkung ist leider auch in der neuen Version 2.10a enthalten: Parameter vom Typ Open ARRAY (offene Arrays, die keine Begrenzung des Indexbereichs aufweisen) müssen als VAR-Parameter deklariert werden. Absurderweise dürfen dennoch Stringkonstanten eingesetzt werden, was bei Unachtsamkeit zu äußerst merkwürdigen Erscheinungen führt. Neben INTEGER und CARDINAL sind auch die 32-Bit-Typen LONGINT und LONGCARD mit einem Wertebereich von 4294967295 zulässig. Der Typ LONGREAL ist dagegen noch nicht implementiert, jedoch bereits vorgesehen. In einem der Bibliotheksmodule ist ein Zufallsgenerator als Funktion mit dem Ergebnistyp LONGCARD vorgesehen.

Der Compiler hat eine fast einzigartige Eigenschaft: Er ist der einzige Modula-Compiler, der keine bekannten Fehler besitzt. Es gibt einige schwer verständliche Fehlermeldungen, zum Beispiel, wenn man eine Funktionsprozedur, die keine Parameter besitzt, ohne die leeren Klammern aufruft. Dann erscheint eine irritierende Meldung über einen fehlerhaften Datentyp, die daher rührt, daß »Funktion« eine Prozedurvariable (und somit implizit den Typ ADDRESS) bezeichnet.

Der Compiler stammt vom Mehrpaß-Compiler der ETH-Zürich, man kann bis zu fünf aufeinanderfolgende Pässe beobachten. Trotzdem arbeitet er mit einer RAM-Disk oder der Harddisk einigermaßen schnell. Mit einem reinen Floppy-System muß man schon etwas Geduld haben. Mit nur einem Laufwerk tut man sich hart, zumindest eine RAM-Disk auf einem Mega-Atari sollte da schon vorhanden sein.

Der compilierte Code ist gut, wenn auch nicht von überragender Güte, was die Geschwindigkeit betrifft. Sehr eindrucksvoll ist die Arbeitsgeschwindigkeit, die ich mit dem jüngst programmierten Programmeditor TLGE erzielte: Er schlägt alle mir bekannten kommerziellen Editoren auf dem ST deutlich sowohl beim Laden, Speichern und beim Springen über große Distanzen im Text. Auch beim Suchen und Kopieren arbeitet er selbst mit großen Textdateien schnell. Er findet ein Wort von etwa 10 Buchstaben am Ende eines 30 KByte großen Textes in drei Sekunden.

Die Bibliotheken sind reichlich ausgestattet

Die mitgelieferten Bibliotheken sind so zahlreich, daß ich hier nur einen Überblick der wichtigsten geben kann. Das Handbuch bietet zwar eine hinreichende Anleitung zur Bedienung des Systems (die inzwischen sowieso kinderleicht ist), verlangt aber Vorkennt-

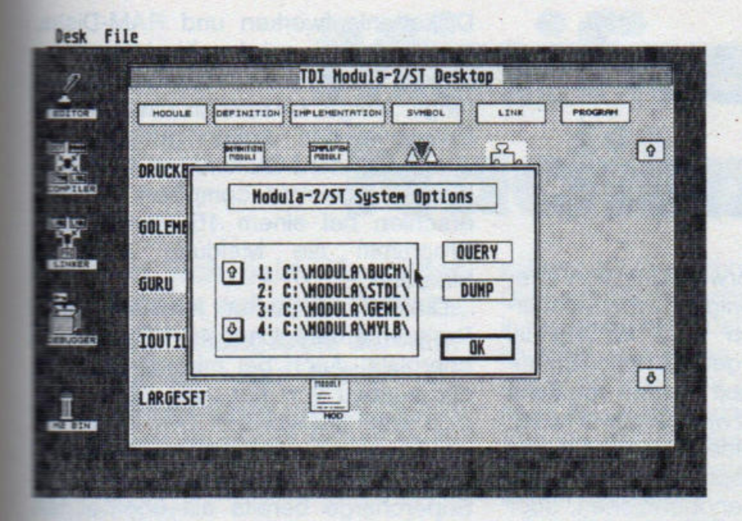


Bild 3. Beispiel für vier verschiedene Pfadnamen

nisse in Modula beziehungsweise gute Kenntnisse der Konzepte moderner Sprachen.

Man kann die Bibliotheken in folgende Gruppen einteilen: – AES-Module (Aplication Environment Support) für die Programmierung unter GEM. Darunter finden Sie zum Beispiel Routinen zur Darstellung eines Gummiband-Rechtecks (Rubber-Box) und verschiedener anderer Grafikelemente von GEM einschließlich der Programmierung mit Fenstern und Resource-Files. – BIOS-Module mit Aufrufen von BIOS-Routinen sowie XBIOS (erweitertes BIOS), mit Prozeduren zur Programmierung des Musik-Chips oder des Tastatur-/Uhrenchips. – GEMDOS-Module zur Programmierung auf TOS-Ebene (Zeichen lesen/schreiben von Files, Tastatur, Drucker- oder Modemport, blockweises Lesen/Schreiben auf Files).

Konventionelle Standardmodule in Anlehnung an N. Wirths Standardwerk für allgemeines I/O (Zeichen, Strings, verschiedene numerische Typen). Auffallend gut ist die Ausstattung mit Konversionen zwischen verschiedenen numerischen Typen einschließlich Oktal, Hex und String.

Die Module orientieren sich entweder sklavisch an den C-Schnittstellen von GEM und TOS, oder sie spiegeln die recht eigenwillige Schöpferkraft der Autoren wider. Viele der Module sind nicht besonders brauchbar, und in der Regel wird man nur wenige davon benutzen. Manche sind auch fehlerhaft beziehungsweise verwenden zweifelhafte Konventionen. So fügt TextIO, wenn man es zur Eingabe von Filenamen durch den Benutzer einsetzen will, zur Korrektur getippte Backspace-Zeichen in den String ein. Hier hätten mehr Sorgfalt und Besinnung besser getan als die wahllose Fülle.

Der Effekt ist jedenfalls, daß ich lieber meine eigenen Module zur Texteingabe, Stringverwaltung etc. von früher verwendeten Modula-Systemen übernehme und die TDI-Module weitgehend ignoriere – was natürlich nicht im Sinne einer Standardisierung (außer der persönlichen) geschieht.

Recht unpraktisch ist die Verwaltung des Heaps und der Files. Man kann nicht abfragen, wieviel Speicher für das Erzeugen von Datenobjekten noch verfügbar ist, sondern nur, ob das Erzeugen gelungen ist oder nicht. Ebenso muß die maximal mögliche Größe des Heaps (soll er über eine bescheidene Standardgröße hinausgehen) per Versuch und Irrtum ermittelt werden, weil man nicht feststellen kann, wieviel Speicher im System zur Verfügung steht (das hängt davon ab, wieviel RAM im Rechner installiert ist, ob RAM- oder ROM-TOS läuft, welche Accessories geladen sind und wieviel Speicher eventuell von einer RAM-Disk belegt wurde). Immerhin läßt sich das in einer einfachen, wenn auch wenig eleganten WHILE-Schleife erledigen.

Das Modul Streams erlaubt nur das Lesen und Schreiben von 8-, 16- oder 32-Bit-Worten (einschließlich wahlfreiem Zugriff), nicht aber das Schreiben und Lesen von beliebigen Records. Insbesondere fehlt eine Meldung oder Abfragemöglichkeit darüber, wann die Diskette voll ist! Da ist es schon besser, man schreibt sich, aufbauend auf dem Modul GEMDOS, sein eigenes Modul FILES, das geht in Modula durchaus elegant und arbeitet auch sehr viel schneller, wie der TLGE beweist.

Für die GEM-Programmierung wurden die sehr technischen und sehr benutzerfreundlichen Schnittstellen von Digital Research auf Modula abgebildet. Man braucht einige Wochen Einarbeitung, um damit zurechtzukommen. Ein Modul zur Fensterverwaltung nach dem (anerkannten) Vorschlag von N. Wirth soll sich in Vorbereitung befinden. Einstweilen haben wir uns unseren eigenen »Windowhandler« geschrieben, um die Sache zu vereinfachen. Sobald man sich erst einmal damit vertraut gemacht hat, ist es nicht allzu schwer, GEM zu programmieren. Angekündigt, aber bis Redaktionsschluß nicht verfügbar war auch ein Toolkit mit einem eigenen Resource Construction Set (einem grafischen Editor zum Zusammenbasteln von Dialog-Boxen, Pull-down-Menüs und Pictogrammen). Bis dahin bleibt man auf das DR-Entwicklungspaket angewiesen.

TLGE: Gute praktische Eignung

Bei der Entwicklung des Programmeditors TLGE (The Little Golem Editor) habe ich mich mit dem TDI-System recht wohl gefühlt. Es gab keine besonderen Schwierigkeiten. In der Endphase wurden die Compiler-Linker-Zeiten trotz Festplatte etwas lästig (einige Minuten), aber da muß ich mich an die eigene Nase fassen, denn das Hauptmodul ist inzwischen über 1200 Zeilen lang (und gehörte längst in mehrere Module für verschiedene Funktionsgruppen zerlegt). Der TLGE kann mit den Funktionstasten des ST oder mit Wordstar-Controlcodes bedient werden, er verfügt über Such- und Blockbefehle und kann zwei voneinander unabhängige Texte editieren und mit oder ohne Zeilennummern sowie Seitenformatierung drucken. Eigentlich handelt es sich um einen Editor-Baukasten, mit dem sich Editoren nach Maß schneidern oder in Anwendungsprogramme einbauen lassen.

Trotz einiger Kritikpunkte kann ich TDI-Modula sowohl zum Kennenlernen der Sprache, für den Unterricht sowie für professionelle Softwareentwicklungen empfehlen. Es ist in meinen Augen in der Gesamtqualität allen anderen mir bekannten Systemen, insbesondere auf dem IBM-PC, überlegen. Der Preis von 350 Mark für das komplette Paket ist als sehr günstig einzustufen. 1983 habe ich Modula kennengelernt, 1984 habe ich ihm viel Glück gewünscht, 1985 wagte ich zu prognostizieren, daß es sich durchsetzen wird. Jetzt weiß ich, daß es nicht mehr aufzuhalten ist.

(le)

Turbolader für SuperBasic

Supercharge von Digital Precision ist der erste Compiler für SuperBasic. Er bringt Ihren QL auf Touren.

Mit SuperBasic gelang eine stimmige Mischung aus Sprachelementen von Basic, Pascal und Forth. Dadurch bietet es einen umfangreichen Wortschatz, der vielen anderen Programmiersprachen abgeht. Aber die Verarbeitungsgeschwindigkeit enttäuscht.

Neugierig ist man deshalb als QL-Besitzer auf ein Produkt, das angeblich alle Basic-Befehle in Maschinenprogramme übersetzt. Der Lieferumfang umfaßt eine Microdrive-Cartridge mit dem Compiler, ein Handbuch von über 100 Seiten und »Lenslok«, die optische Sicherung gegen Schwarzkopien.

Diese Sicherung entpuppt sich allerdings oft als Ärgernis. Denn selbst als Besitzer des Originals scheitert man nicht selten daran – und eine Menge Zeit nimmt sie obendrein in Anspruch.

Mehr Tempo braucht massenhaft Platz

Läßt man dies jedoch außer Betracht, so kann Supercharge nur begeistern.

Der Compiler ist nicht nur sehr schnell, sondern produziert auch schnellen Code. Geschwindigkeitserhöhungen um den Faktor 60 sind keine Seltenheit. Bei Verwendung von Integerzahlen und einigen Programmiertricks, von denen viele im Manual beschrieben sind, geht es noch schneller. Im Programmtext besteht von Zeile zu Zeile die Wahl zwischen zwei Modi. Entweder entscheidet man sich für eine Geschwindigkeitsoptimierung, zum Beispiel bei Rechenoperationen, oder man zieht kürzeren Code vor. Aber Achtung, ein 6-KByte-Maschinenprogramm kann bei größerer Geschwindigkeit leicht auf 30 KByte anwachsen!

Darüber hinaus wird der Programmierer zu konsequentem Programmstil erzogen. Während des Compiliervorgangs »verwarnt« das Programm über den Drucker, sobald ein schlechter Stil vorliegt. Es resultieren zwar keine Programmfehler daraus; wenn man diese Warnungen aber beachtet, zahlt es sich durch eine bessere Überschaubarkeit und weniger Zeitaufwand bei späteren Änderungen aus.

Supercharge verwendet bei der Ausgabe von Zahlen eine größere Genauigkeit als der SuperBasic-Interpreter. Das intern bereits hohe Darstellungsformat wiederholt sich auch bei der Ausgabe. Ein weiterer wichtiger Aspekt von Supercharge ist die Fähigkeit, ehemalige SuperBasic-Programme simultan durch das Multitasking laufen zu lassen. Das erlaubt das QL-Betriebssystem QDOS bei allen Programmen, die man als Maschinencode-Programm mit dem Kommando EXEC aufruft. Supercharge verhält sich auch kompatibel zu allen Diskettenlaufwerken und RAM-Disks, die für den QL auf dem Markt sind.

Beim Compilieren traten im Test allerdings einige Unzulänglichkeiten auf. Entgegen den Ankündigungen, auf einem Standard-QL könnten 40 KByte lange Programme compiliert werden, erschien bei einem 15-KByte-Basic-Programm die Meldung »Out of Memory«.

Dieses ansonsten leistungsstarke Programm belegt nur 47 KByte Speicherplatz. Auch bei minimalem Speicherausbau von nur 128 KByte läßt sich damit gut arbeiten. Dies zeigt zum einen, wie gut der Compiler übersetzt, denn nach Herstellerangaben ist Supercharge bereits ein übersetztes Basic-Programm von 2500 Zeilen. Zum anderen verdankt man dies dem komfortablen Basic-Interpreter, dessen Tabellenstruktur ein Compiler mitverwenden und schnell durchlaufen kann.

Cooperativer Interpreter

Das Handbuch erklärt sehr ausführlich alle Details und erläutert außerdem noch global die Arbeitsweise des Compilers. Wer sich dafür interessiert, findet interessante Hinweise.

Im Test erhielt Supercharge eine gute Bewertung. Die erwartete Geschwindigkeitssteigerung übertrifft der Compiler sogar bei weitem. Die Wermutstropfen »Lenslok« und die geringe Speicherkapazität für die Basic-Programme in der Grundversion trüben den eitlen Sonnenschein ein wenig. Aber bei weiterhin fallenden Preisen für RAM-Karten wird zumindest letzteres nicht mehr lange ein Thema sein.

(Harald Wilhelm/hb)







Minifinder mit Nachbrenner

Befinden sich beim Macintosh viele Dateien auf einer Diskette, dauert es oft lange, bis der »Finder« wieder »klar Schiff« auf dem Schreibtisch gemacht hat. Außerdem benötigt er auf jeder Diskette 47 KByte. Nicht so der »Minifinder«.

Der Minifinder hat seit der Version 4.1 seinen Platz im Finder, aber trotz seiner vielen Vorteile wird dieses Programm relativ selten genutzt. Der Minifinder erlaubt es, bis zu zwölf Programme beziehungsweise Dateien gleichzeitig auf dem Schreibtisch abzubilden. Will man mehr Programme öffnen, ist der Befehl »Andere öffnen« anzuklicken. Dies aktiviert eine Dialogbox, von der aus alle Dateien und Programme auf der Diskette geöffnet werden können.

Der Finder in seiner bekannten Form ist damit überflüssig. Tatsächlich gibt es eine Reihe von Macintosh-Benutzern, die ohne Finder-Datei arbeiten. Das Verfahren hat den Vorteil, daß auf einer solchermaßen konfigurierten Diskette gut 40 KByte mehr Platz für Daten zur Verfügung steht, da die Minifinder-Datei nur rund 5 KByte Platz beansprucht.

Um den Finder loszuwerden, geht man folgendermaßen vor: Zunächst richtet man den Minifinder entsprechend den Anweisungen im Handbuch mit den gewünschten Dateien und Programmen ein. Binnen Sekunden erscheint auf dem Schreibtisch eine neue Datei mit der Bezeichnung »Minifinder«, in der festgehalten ist, welche Symbole im Minifinder laufen sollen. Jetzt muß der Macintosh neu gestartet werden, aber mit einer anderen Startdiskette. Nun wandert die Finder-Datei auf der Diskette mit dem installierten Minifinder in den Papierkorb.

Für den Fall, daß die Beschränkung auf zwölf Dateien und Programme im Minifinder den Arbeitsablauf zu stark behindert, lassen sich mehrere Minifinder ineinander verschachteln. So ist es möglich, in einen Minifinder bis zu zwölf Unter-Minifinder einzubauen, die ihrerseits jeweils bis zu zwölf Dateien oder Programme enthalten. Die beiden Ebenen reichen damit aus, bis zu 144 Objekte zu verwalten.



Wenn ein Dutzend zu wenig ist

Der Minifinder in der obersten Ebene enthält dann zwölf Minifinder-Symbole, die jedoch neue Bezeichnungen benötigen. Ein Doppelklick auf eines dieser Symbole öffnet den zugehörigen Unter-Minifinder, der neben elf Programmen zweckmäßigerweise ein weiteres Minifinder-Symbol enthält. Dieses ist erforderlich, wenn man wieder zum Minifinder der obersten Ebene zurückkehren will. Hier nun die Schritte, mit denen man einen solchen Minifinder-Schreibtisch zusammenstellt. Die zweite, untere Ebene richtet man dabei vor der oberen, ersten ein.

Erster Schritt: Zunächst kopiert man alle Dateien oder Programme, die im Minifinder Platz finden sollen, in einen Ordner. Wünschen Sie, von dieser untergeordneten Minifinder-Ebene zurück zur oberen springen zu können, muß sich in diesem Ordner auch ein Minifinder-Symbol mit der Bezeichnung »Minifinder« befinden. Es dient als Platzhalter für den Minifinder der oberen Ebene, der später installiert wird.

Zweiter Schritt: Um diesen Platzhalter zu schaffen, aktiviert man ein beliebiges Programm und wählt aus dem Menü »Spezial« den Befehl »Minifinder installieren«. Sekunden später erscheint ein Symbol mit der Bezeichnung »Minifinder«, das nur noch in den richtigen Ordner zu ziehen ist.

Dritter Schritt: Nun installiert man nacheinander alle Minifinder der zweiten, unteren Ebene nach dem gleichen Verfahren. Wichtig: Alle Minifinder der zweiten Ebene müssen unterschiedliche Namen aufweisen, da sonst die ganze Sache nicht funktioniert. Verzweifeln Sie aber nicht gleich, wenn die ersten Umbenennungsversuche vom Macintosh mit Fehlermeldungen quittiert werden. Ein kleiner Trick schafft da Abhilfe. Aktivieren Sie zunächst Ihre elf Programme sowie den Platzhalter-Minifinder. Wählen Sie dann aus dem Menü »Spezial« wieder den Befehl »Minifinder installieren«, worauf ein neues Minifinder-Symbol erscheint. Dieses aktivieren Sie und wählen aus dem Menü »Datei« den Befehl »Duplizieren« aus. Schneller geht dies natürlich mit der Tastenkombination Befehlstaste-D. Darauf erscheint ein neues Symbol mit der Bezeichnung »Kopie von Minifinder«, das man nun umbenennen kann. Ist dies geschehen, wandert der zunächst generierte Minifinder (von dem die später umbenannte Kopie gezogen wurde) in den Abfalleimer.

Vierter Schritt: Nach dem beschriebenen Verfahren benennen Sie nacheinander alle Minifinder um.

Fünfter Schritt: Um den Minifinder der obersten Ebene zu installieren, werden alle (umbenannten) Minifinder der zweiten Ebene aktiviert. Dann wählt man aus dem Menü »Spezial« erneut den Befehl »Minifinder installieren« und versetzt nach dem Erscheinen des Minifinder-Symbols mit einem Doppelklick den Minifinder der obersten Ebene in Tätigkeit. Dieser erscheint übrigens auch bei jedem Neustart der Diskette, auf der er sich befindet.

Bei Platzmangel kann man nun die Finder-Datei auf der Diskette löschen. Sie wird nicht mehr benötigt.

Das Verfahren beschleunigt die Arbeit mit dem Macintosh enorm. Vor allem bei vielen Dateien kommt man so auch ohne hierarchisches Dateisystem zu einer übersichtlichen, nach Sachgruppen geordneten Datenverwaltung, die zudem noch zeitsparender arbeitet als mit dem herkömmlichen Finder.

(Reto Tanner/ts)

Hierarchie auf

»Wer Ordnung wahrt, der Arbeit spart«, sagt schon das Sprichwort. Ein neues Dateiverwaltungs-Konzept sorgt nun endlich auch beim Macintosh dafür, daß die Übersicht selbst bei großen Datenmengen nicht verlorengeht.

Seitdem die Uhr zum ersten Mal auf dem Bildschirm des Macintosh erschien, wurde bei vielen Anwendern der Wunsch nach schnelleren Diskettenlaufwerken mit höherer Speicherkapazität und einem Betriebssystem laut, das auch viele Dateien übersichtlich und zeitsparend verwalten kann.

Ab sofort ist beides verfügbar: Doppelseitig formatierte 3,5-Zoll-Floppies speichern 800 KByte Daten und die Finder-Betriebssystem-Version 5.1 erschließt auch Macintosh-Anwendern die Welt der hierarchischen Dateiverwaltung. Apple spricht in diesem Zusammenhang vom HFS; die Abkürzung steht für »Hierarchical File System«, im Gegensatz zum MFS, dem »Macintosh File System«.

Zuviel des Guten

Es war zu erwarten, daß das Dateiverwaltungssystem MFS bei einer Vergrößerung der Speicherkapazität der Disketten hoffnungslos überfordert sein würde. Schließlich sollte es ursprünglich nur den Zweck erfüllen, Dateien in einer Zahl, wie man sie bequem auf 400-KByte-Floppies unterbringen kann, zu kopieren, zu verschieben, neu zusammenzustellen oder umzubenennen. Eine derart rasante Kapazitätssteigerung bei den Macintosh-Diskettenlaufwerken überraschte aber allem Anschein nach das Team der Macintosh-Entwickler völlig.

Nun geht es bei einer mit vielen kleinen Dateien voll belegten 800-KByte-Diskette noch an, unter dem herkömmlichen Macintosh-Dateiverwaltungs-System vom Finder aus eine gewünschte Datei zu suchen. Vom Programm aus den Befehl »Öffnen« anzuklicken, um dann womöglich unter Hunderten von Dateien eine bestimmte herauszusuchen, stellt sich jedoch als ein sehr mühsames Unterfangen heraus und ist damit für den professionellen Einsatz ungeeignet. Die Verdoppelung der Speicherkapazität bei den neuen Laufwerken macht die Grenzen des MFS bei der Verwaltung vieler Dateien überdeutlich.

Mehrere Hersteller von Festplatten-Systemen für den Mac umgingen diese Situation durch eine Aufteilung des verfügbaren Speicherplatzes in verschiedene »Volumes«. Die Festplatte verhält sich dabei wie mehrere einzelne Disketten. Nachteilig ist, daß man in aller Regel nicht auf alle Volumes gleichzeitig zurückgreifen kann. Die Festplatte führt also nur die Funktion eines einheitlichen, aber in eine Vielzahl kleinerer Speichereinheiten unterteilten Massenspeichers aus. Zusätzlich verändern die Volumes ihre Größe nur vereinzelt dynamisch, also den anfallenden Datenmengen entsprechend. Außerdem wird beim Einrichten eines Volumes meist ein fixer »minimaler« Speicherplatz reserviert. Braucht man ihn nicht, ist er vergeudet. Reicht der dem Volume zugewiesene Platz zum Speichern einer Datei jedoch nicht aus, kann man dessen Größe andererseits nicht »dehnen«, sondern muß ein zusätzliches einrichten. Manche Festplattensysteme installieren darüber hinaus für jede Datei (unabhängig von deren tatsächlicher Größe) einen festen minimalen Speicherplatz, mit der Folge, daß bisweilen 10-KByte-Speicherplätze nur mit 2-KByte- oder 3-KByte-Dateien belegt sind, der Rest bleibt ungenutzt.

Diese und noch andere Nachteile zu umgehen, ist der Zweck des neuen hierarchischen Dateiverwaltungs-Systems namens HFS, das Apple mit dem MacPlus und der Festplatte HD20 einführte. Bei der HD20 noch in Form einer Startdiskette, ist es im MacPlus fest im 128-KByte-ROM untergebracht.

Die Art und Weise, wie beim HFS Dateien und Anwenderprogramme auf Disketten oder Festplatten verwaltet werden, entspricht im wesentlichen dem System der Archivierung von Unterlagen in Aktenordnern und Aktenschränken. Einziger Unterschied: Beim HFS können mehrere Ordner ineinander verschachtelt werden. Auf einer Diskette ist beispielsweise ein Ordner für Manuskripte angelegt, der wiederum selbst in Ordner für die verschiedenen Computersysteme unterteilt ist.

Wie viele Unterteilungs-Ebenen man mit dem HFS anlegt, bleibt jedem Anwender selbst überlassen und orientiert sich vornehmlich daran, wie stark man im Einzelfall seine Dateien unterscheiden muß. Um einen bestimmten Ordner zu öffnen oder zu schließen, steht einem sowohl der Weg über den Finder/Schreibtisch als auch über die Dialogboxen innerhalb eines Programms unter den Befehlen »Öffnen« oder »Sichern als« zur Verfügung.

Achtung: Verwechslungsgefahr!

Versierte Macintosh-Benutzer stolpern beim HFS erfahrungsgemäß vor allem über folgende Punkte: Wer Wert darauf legte, konnte auch bislang schon Ordner ineinander verschachteln. Eine Untergliederung nach dem Motto, »Von der allgemeineren Bezeichnung zur immer differenzierteren« war, rein formal betrachtet, schon immer möglich. So fragt man sich an dieser Stelle, warum es Apple nicht verstanden hat, anstelle der von den MFS-Ordnern nicht zu unterscheidenden HFS-Ordner ein neues Symbol einzuführen (denkbar wäre zum Beispiel eine stilisierte Schublade), so daß beim Arbeiten unter dem Finder (= Schreibtischebene) bereits eindeutig klar wird, ob diese unter HFS oder MFS läuft. Jetzt sehen die Ordner in beiden Fällen gleich aus. So läuft man Gefahr, daß eine vermeintlich unter MFS installierte Diskette tatsächlich unter HFS läuft und damit von einem nicht aufgerüsteten Macintosh mit 512 KByte beziehungsweise 128 KByte nicht akzeptiert wird. Demgegenüber erkennt jedoch ein MacPlus oder ein aufgerüsteter 512-KByte-Mac eine unter MFS initialisierte Diskette sehr wohl.

Der feine Unterschied zwischen dem »alten« MFS und dem neuen hierarchischen Dateiverwaltungssystem HFS wird sichtbar, wenn beim Aufruf einer der Funktionen »Öffnen«, »Sichern« oder »Sichern als« die dazugehörigen Dialogboxen auf dem Bildschirm erscheinen. Während der MFS im Rollfeld alle auf der Diskette gespeicherten

dem Macintosh

Dateien auflistet, zeigen die Dialogboxen beim HFS in ihrem Rollfeld ein Verzeichnis der Ordner auf der Diskette. Wählt man davon einen an, so erscheint dessen Inhaltsverzeichnis. Das können nun Dateien oder, falls eine weitere Untergliederung vorgenommen wurde, andere Ordner sein.

Neu beim HFS kommt hinzu, daß im Rollfeld der Dialogbox zusätzlich zu den einzelnen Bezeichnungen mit kleinen Symbolen angezeigt wird, ob es sich um einen Ordner oder um eine Datei handelt. Die Bezeichnung der Diskette beziehungsweise des aktiven Ordners, dessen Inhalt das Rollfeld gerade zeigt, ist in einem Feld darüber abzulesen. Falls man nicht mehr weiß, in welchem »Stockwerk« ineinandergeschachtelter Ordner man sich befindet, aktiviert man das Feld, so daß es wie ein Rollmenü herunterfällt und die verschiedenen Ebenen zeigt. Analog zur Vorgehensweise beim Rollmenü ist damit ein Rücksprung in eine übergeordnete Hierarchie-Ebene möglich.

Versteckspiel mit Dateien

Die Auflistung von Dateien und Ordnern im Rollfeld erfolgt in alphabetischer Reihenfolge. Wünschenswert wäre eine Möglichkeit, sie in Sachgruppen oder chronologisch geordnet sehen zu können, um so das Auffinden noch weiter zu vereinfachen. Unter dem Befehl »Öffnen« erscheinen im Rollfeld nur jene Dateien, die vom laufenden Programm aus aufgerufen werden können. Dies vereinfacht zwar die Suche nach einer bestimmten Datei, erschwert aber gleichzeitig die Arbeit, wenn zum Beispiel von MacWrite aus festgestellt werden muß, ob eine MacPaint-Datei auf der aktiven Diskette gespeichert wurde.

Bequem ist das Speichern beim HFS mit den Dialogboxen unter den Befehlen »Sichern« und »Sichern als«. Sie erlauben es, eine Datei gleich in einen gewünschten Ordner (und nicht mehr nur auf der Schreibtischoberfläche) zu packen.

Eine weitere Erleichterung im Umgang mit Dateien unter HFS ist es, die Bezeichnung einer zu öffnenden Datei über die Tastatur eingeben zu können, sobald der Befehl »Öffnen« erteilt wurde. Prinzipiell genügt die Eingabe so vieler Zeichen, daß eine einwandfreie Unterscheidung zwischen zwei ähnlich bezeichneten Dateien vom HFS gewährleistet ist. Bei der Eingabe ist darauf zu achten, daß zwischen zwei Zeichen nicht zu viel Zeit verstreicht, weil das System längere Pausen als Ende der Eingabe interpretiert.

Diese Eigenschaft der hierarchischen Dateiverwaltung ist ganz besonders gewöhnungsbedürftig, zumal kaum ein anderes Betriebssystem diese Funktion kennt. Es fragt sich, ob die übliche Bestätigung mit einem Druck auf die RETURN-Taste nicht auch gereicht hätte. Doch Apple ist bekanntlich immer für eine Innovation zu haben, und sei sie noch so ausgefallen.

HFS ist nicht nur ein Mittel, eine Vielzahl von Dateien übersichtlich zu ordnen. Die hierarchische Struktur erlaubt es auch erstmals, in verschiedenen Ordnern auf einer Diskette Dateien mit gleicher Bezeichnung zu speichern.

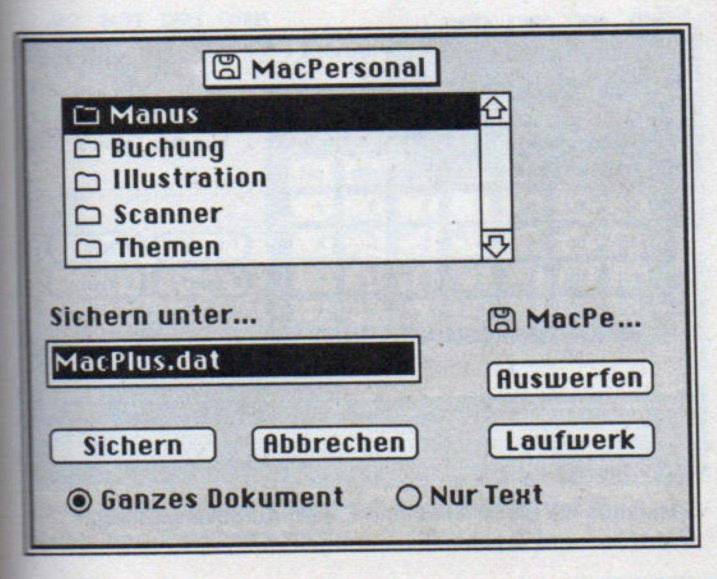
Nach einer kurzen Eingewöhnungszeit ist das hierarchische Dateiverwaltungssystem recht einfach und komfortabel zu handhaben. Falls überhaupt, haben nur Anwender, die auf das MFS-System eingefahren sind, in der ersten Zeit leichte Anpassungsschwierigkeiten. Denn Apple hat keine visuelle Unterscheidungsmöglichkeit zwischen beiden Systemen geschaffen. Die Vermutung, daß das Öffnen stark ineinander verschachtelter Ordner auf Dauer ziemlich frustrierend sein würde, hat sich nicht bestätigt. Wenn man sich gleich am Anfang bemüht, die vorhandenen Dateien nach einem übersichtlichen und leicht erweiterbaren »Suchbaum« zu ordnen, kommt man schnell mit der hierarchischen Dateiverwaltung zurecht.

Nützliche Erweiterungen

Die neue Finder-Version 5.1 bietet zudem zwei neue Befehle. Unter »Datei« findet man den Befehl »Zurücklegen«, mit dessen Hilfe Dateien oder Programme vom Schreibtisch zurück in den Ordner befördert werden, in dem sie sich zuletzt befanden. Unter »Abbildung« erscheint der Befehl »Kleine Symbole«, der eine Miniaturdarstellung der Symbole auf dem Bildschirm bewirkt, was bei einer großen Anzahl von Dateien vorteilhaft ist.

Wünschenswert wäre es gewesen, neue Befehle in den Finder zu integrieren, die ein Arbeiten mit einer Vielzahl von Dateien erleichtern. Wurde eine Datei in einen falschen Ordner abgelegt, ist es nicht leicht, sie später wieder zu finden. Hier wünscht man sich eine Suchmöglichkeit unter allen Ordnern durch Eingabe der Dateibezeichnung. Interessant wäre auch, eine strukturierte Übersicht der Ordner und der Dateien auf den Bildschirm holen beziehungsweise ausdrucken zu können.

(Peter Kraft/ts)



Die »Sichern als«-Dialogbox mit verschiedenen Ordnern. Ein Anklicken des Ordners zeigt die darin enthaltenen Dateien.

Heinzelmännchen für den ST

Nützliche Hilfsprogramme für die tägliche Arbeit lassen sich mit dem Desktop aus anderen GEM-Programmen heraus aufrufen. Wir haben solche Heinzelmännchen-Programme getestet.

Im umfangreichen Speicher des Atari ST kann man eine ganze Wundertüte voll nützlicher Routinen verstecken, für die im Desktop das linke Pull-down-Menü reserviert wurde. Leider lassen sie sich nur aus einem GEM-Programm hervorlocken.

»Calendar« entwickelte das amerikanische Softwarehaus Michtron. Es kombiniert einen komfortablen Kalender mit integrierter Terminverwaltung. Wie alle Accessories muß es sich beim Booten des Betriebssystems auf der Diskette im Laufwerk A befinden. Nach dem Booten des Accessory meldet sich »Calendar« mit allen für diesen Tag erfaßten Terminen und gibt so schon zu Beginn der Arbeit einen guten Überblick über die Termine des Tages. Will man einen neuen Termin eingeben oder ändern, wählt man zuerst den betreffenden Monat und danach den richtigen Tag, und schon erscheint das Eingabemenü. 24 Stunden eines Tage können Sie nun in Viertelstunden-Einteilung mit Terminen belegen. Etwas ungewohnt wirkt dabei allerdings die Uhrzeitangabe amerikanischen Formats. Eine Anpassung an deutsche Verhältnisse ist nicht geplant.

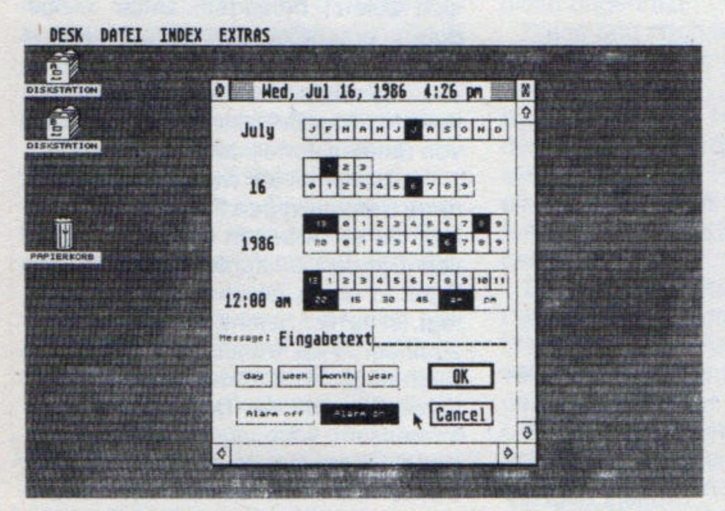
Klickt man den oberen rechten Fensterpunkt an, so gelangt man in ein weiteres Menü mit folgenden Punkten: Ausdruck des ganzen Tages, Erinnerung an den eingegebenen Termin im Stundenrhythmus und die Funktion »DIRECT ENTRY«. Mit ihr hat es folgende Bewandtnis: Will man einen Dauertemin eingeben, so erübrigt es sich, einen neuen Monat auszuwählen oder gar ein neues Jahr, um dann wieder in das Eingabemenü zu verzweigen. Im »DIRECT ENTRY« kann man auf einfachste Weise Tage, Wochen, Monate und Jahre vor- und zurückspringen oder einen Termin über Datumseingabe an den richtigen Tag adressieren. Das spart Zeit. Die Eingabe von Umlauten quittiert »Calendar« mit fehlerhaftem Speichern auf Diskette. Leider fehlt diesem Programm eine Suchfunktion. Der Termin mit Herrn Müller schlummert so eventuell auf ewig im Speicher. Es sei denn, man durchforstet den in Frage kommenden Bereich Datum für Datum ab. Neben dem Ausdruck eines einzelnen Tages gestattet dieses Accessory auch, einen bestimmten Monat oder ein ganzes Jahr ausdrucken zu lassen. Allerdings nicht mit den eingetragenen Terminen, sondern nur mit den normalen Kalenderdaten.

Blue Moon Software vertreibt das Accessory »MacroDesk«, das eine ganze Anzahl von Funktionen vereint: einen Taschenrechner, eine Adreßverwaltung, eine Alarmbox und eine Art Tagebuch.

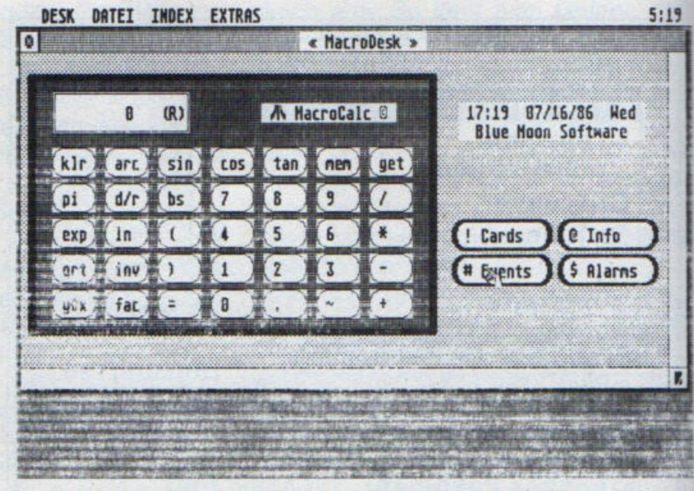
Der Taschenrechner beherrscht alle wichtigen Grundrechenfunktionen inklusive Tangens und Sinus. Auch eine Speicherfunktion wurde integriert. Die Zahleneingabe erfolgt über die Tastatur oder den Zehnerblock. Leider dient die Return- und Enter-Taste nicht dem Abschluß einer Rechenoperation. Man muß jedesmal mit der Maus das Gleichheitszeichen anklicken oder die betreffende Taste drücken. Für den Hausgebrauch reichen die Funktionen des Rechners aus.

»MacroDesk« – ein Tisch voller Utilities

Im zweiten Modul sitzt eine Adreßverwaltung, die zwar recht spartanisch aufgebaut, aber funktionsfähig und kinderleicht zu bedienen ist. Bis auf Name und Telefon gibt es keine vorbestimmten Felder, so daß man Ort, Land und Straße nach Belieben eintragen kann. Ist ein Datensatz erfaßt, speichert das Programm automatisch das Datum und die Zeit der Erfassung. Man kann Datensätze beliebig hinzufügen, löschen und



»Calendar« kombiniert einen Kalender mit umfangreicher Terminverwaltung



»MacroDesk« bietet Taschenrechner, Adreßverwaltung, Alarmbox und Tagebuch

ändern. Der Ausdruck erfolgt über vier vorbestimmte Masken, zum Beispiel für Adreßaufkleber oder Adreßlisten. Die Suchfunktion ist schlicht unkomfortabel. »Macrodesk« zeigt die Datensätze nur sequentiell auf dem Bildschirm an. Suchstrings nimmt es nicht an. Dafür wartet es mit einer Funktion auf, die für amerikanische Verhältnisse selbstverständlich ist: »Auto Dial«. Mit »Auto Dial« kann man eine gespeicherte Telefonnummer auf Tastendruck an ein Modem ausgeben und vom Computer eine Telefonverbindung herstellen lassen. Die elektrische Sekretärin ist Wirklichkeit! Da in Deutschland die Post den Betrieb eines Modems mit Selbstwählvorrichtung nicht gestattet, bleibt diese Funktion für uns leider nutzlos.

Das Tagebuch gleicht optisch und auch von der Funktionsweise der Adreßdatei und dient nur der Unterteilung in zwei Dateien.

Sehr zuverlässig arbeitet die Alarmbox. Man trägt den Termin und eine Erinnerungsmitteilung ein und kann sicher sein, zu rechten Zeit optisch und akustisch auf diesen Termin hingewiesen zu werden.

»Powerpack« – drei auf einen Streich

Im Gegensatz zu »Macrodesk« ist »Powerpack« aus dem Haus Blue Moon Software kein Multiaccessory. »Powerpack« bietet eine Konstellation von Adreßdatei, freier Dateiverwaltung und Terminverwaltung, allerdings als Einzelaccessoires. Dies wirkt sich nachteilig aus, da »Powerpack« mehr Platz als nötig im Pull-down-Menü der Accessoires beansprucht. Andererseits kann man dadurch auch eine Funktion desaktivieren, wenn man sie nicht braucht. Die Adreßverwaltung arbeitet mit bis zu neun Dateien. Hat man eine Datei ausgewählt, klickt man aus einem weiteren Menü den entsprechenden Anfangsbuchstaben an und befindet sich in der Eingabemaske seiner Datei. Die Eingabemaske der Adreßdatei wurde fest vorgegeben, enthält aber genügend Platz, um alle notwendigen Angaben zu einer Adresse unterzubringen. Benötigt man einen bestimmten Namen, so bietet »Powerpack« eine Suchfunktion. Man gibt nur den Suchbegriff ein, und sofort erscheint die gesamte Adresse auf dem Bildschirm. Da die Eingabemaske wie ein normales Fenster aufgebaut ist, gestatten die Pfeiltasten ein »Blättern«.

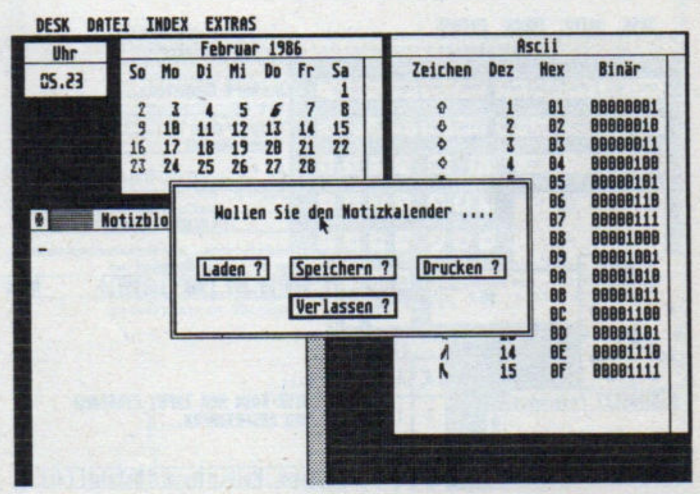
Nun zur Terminverwaltung. Wird dieser Teil von »Powerpack« mitgebootet, so erscheint in der rechten Ecke der Menüzeile eine Digitaluhr, die die im Kontrollfeld eingestellte Uhrzeit anzeigt. Auch das Unterprogramm »Diary« erlaubt, aus mehreren Dateien für verschiedene Jahre zu wählen. Nach Eingabe von Jahr, Monat und Tag befindet man sich in der Eingabemaske, die den Tag in bis zu zehn Bereiche unterteilt. Diese Beschränkung führt bei einem gut gefüllten Terminplan eventuell zu Platzproblemen. Wie bei »Calendar« hält sich die Zeitangabe an das amerikanische Vorbild. Es unterscheidet nach »am« (vor 12 Uhr) und »pm« (nach 12 Uhr). Die Suchfunktion arbeitet wie bei der Adreßdatei.

Fünf Minuten vor einem Termin meldet sich »Powerpack« in der oberen rechten Bildschirmecke mit einem Glockensymbol und einem akustischen Signal. Natürlich auch während der Bearbeitung eines anderen Programms. Hat man den Ton des Monitors sehr leise gestellt, fällt die Erinnerung allerdings nicht sehr deutlich aus, da das kleine Glockensymbol leicht zu übersehen ist. Ein größeres Fenster hätte hier bessere Dienste geleistet. Gut zu handhaben ist die frei definierbare Dateiverwaltung. Sie erzeugt bis zu neun unterschiedliche Dateien, deren Eingabemaske nicht vordefiniert ist. Man kann also neun wirklich unterschiedliche Karteien aufbauen, die sehr gut über die verschiedensten Daten Auskunft geben. Auch erleichtert die gute Suchfunktion das schnelle Auffinden von Begriffen. Ein Mangel des Programms mindert den sonst guten Gesamteindruck immens: Er erlaubt keinen Ausdruck von Dateien. Für den Einsatz im Büro eignet sich das Programm dadurch nicht, denn wer will schon jedesmal eine Hardcopy vom Bildschirm ziehen. Schade!

»Side Click« – ein ganzes Büro im Desktop

Als echtes Multitalent empfiehlt sich »Side Click« von RDS. Die Raunheimer Firma scheute für dieses Programm keine Mühen, und das Ergebnis kann sich sehen lassen. Vom Hauptmenü aus bietet »Side Click« neben Kalender, Terminverwaltung, Taschenrechner, Uhr und Wecker auch Notizblock, Zeichentabelle, Druckeranpassung und eine Funktion zum Ausdrucken von Inhaltsverzeichnissen von Disketten und verbraucht doch nur den Installationsplatz eines Accessories. Anders steht es bei »Side Click« mit dem Speicherplatz. Dieses Accessory verschlingt über 100 KByte.

»Powerpack« ist eine Kombination aus Adreßdatei, freier Daten- und Terminverwaltung



»Side Click«, das Multitalent für den deutschen Anwender besteht aus Teilen

»Side Click« kann das Directory einer beliebigen Diskette nach Namen, Größe, Datum oder Typ geordnet entweder in Normalschrift, Schmalschrift oder in einer speziellen Labelschrift ausdrucken. Letztere Schriftart eignet sich besonders für das Beschriften von Disketten-Etiketten. Die Druckeranpassung wurde gut gelöst. Damit kann man jedem Epson-kompatiblen Drucker die verschiedensten Parameter übermitteln. Über Steuerzeichen sind sogar norwegische Sonderzeichen möglich. Lästiges Aufschrauben und Dip-Schalter-Verstellen entfällt dadurch.

Die Zeichentabelle stellt alle im ST-Zeichensatz vorkommenden Zeichen in einer Tabelle als Hex-, Binär- und Dezimalwerte dar. Dies erleichtert Programmierern die Arbeit ungemein, leistet aber ebenso bei der Druckeranpassung gute Dienste. Der Taschenrechner orientiert sich an den Bedürfnissen von Programmierern und wissenschaftlichen Anwendern. Hier glückte es, einen Sharp-Rechner mit all seinen vielen und leistungsfähigen Rechenoperationen nachzubilden. Werte können als Hex-, Binär- oder Dezimalwerte angezeigt. Rechenoperationen können Sie auf RAD, GRAD und DEG einstellen. Als praktisch erweist sich die Anzeige des Speicherinhaltes oder des letzten Eingabewertes. Neben den Grundrechenfunktionen beherrscht der Rechner arithmetische Rechenoperationen und gestattet den Austausch von X und Y. Braucht man nur die Grundfunktionen, ist der Umgang mit so einem wissenschaftlichen Rechner allerdings etwas aufwendig und bedarf einiger Einarbeitung. Danach aber wird er schnell zu einem unersetzlichen Instrument.

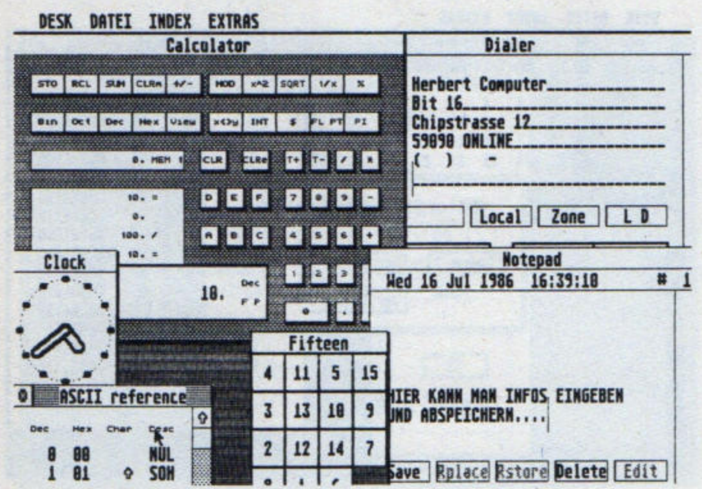
Einen mächtigen Block stellt die Terminverwaltung mit eigenständigem Kalender, Wecker und Uhr dar. Bei der Uhr handelt es sich um eine kleine Digitaluhr, die beliebig auf dem Desktop plaziert die aktuelle Zeit anzeigt. Der Kalender stellt immer einen ganzen Monat dar und hebt Sonntage und den aktuellen Tag hervor. Mit den Fensterschiebern blättert man auf einfachste Weise um Monate vor oder zurück. Die eigentliche Terminverwaltung, die zusätzlich separat auf der Programmdiskette gespeichert ist, verwaltet mehrere, verschiedene Termindateien. Nach Auswahl von Datei, Jahr, Monat und Tag gibt man seine Termine ein. Das Eingabefeld geriet allerdings recht klein, so daß es bei der Verwaltung vieler Termine an einem Tag möglicherweise zu Problemen kommt. Die Termine jeden Tages lassen sich einzeln auf den Bildschirm oder den Drucker ausgeben. Auch eine schnelle Suchfunktion enthält der Terminplaner, der die Datei über zwölf Monate eines ausgewählten Jahres nach jedem beliebigen Suchstring durchgeht und die betreffenden Termine auf dem Bildschirm anzeigt.

»Cornerman« – zu schade, um in der Ecke zu liegen

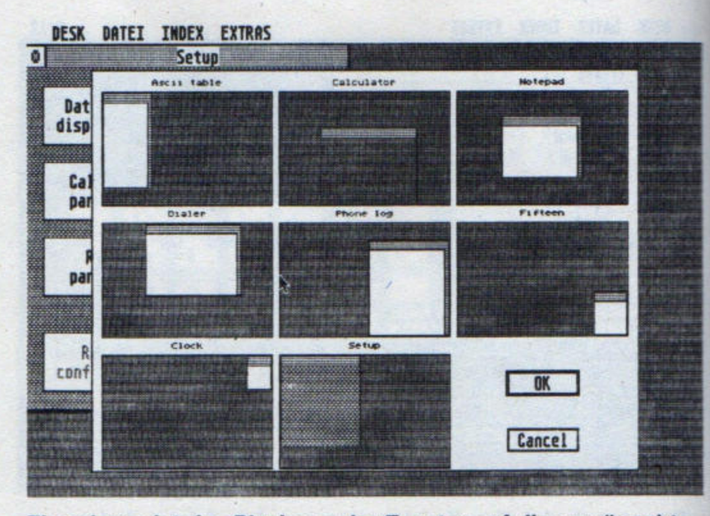
Sehr praktisch und kinderleicht zu bedienen ist die Weckfunktion. Bis zu vier Alarmzeiten lassen sich hier eingeben. Zur rechten Zeit erinnern die Klänge Big Bens und ein Bildschirmfenster an Verabredungen, bis man die Return-Taste betätigt. Speichern lassen sich die Einträge des Weckers nicht. Für manche täglichen Routinearbeiten wäre das eine hervorragende Hilfe. Die neunte und letzte Funktion von »Side Click« ist ein Notizblock mit 20 Seiten. Er dient zum Erfassen, Speichern und Ausdrucken von Notizen und Adressen. Als kleines Bonbon enthält die Programmdiskette noch ein Puzzlespiel als Accessory. Macintosh läßt grüßen. Ein mächtiges Programm, mit kleinen Schönheitsfehlern, die den Nutzen aber kaum beeinträchtigen.

Michtron vertreibt neben »Calendar« auch »Cornerman«. Es verpraßt noch etwas mehr Speicherplatz als »Side-Click«. Dafür bietet es aber auch einiges an Nutzbringendem. »Cornerman« besteht aus: Taschenrechner, Notizblock, Adreßdatei, Uhr, Autodialing, Setup-Menü, ASCII-Zeichentabelle, Drucker-Menü und »DOS WINDOW«. Der Taschenrechner ist sehr umfangreich und doch problemlos zu bedienen. Er umfaßt alle wichtigen Funktionen und läßt sich über Maus, Nummernblock und Funktionstasten steuern. In einem separaten Feld zeigt er die letzten vier Eingaben an. Der Notizblock akzepiert keine Umlaute. Bei der Eingabe führt er automatisch einen »Wordwrap« aus. Das bedeutet, ein Wort, das nicht mehr ganz in die Zeile paßt, wird komplett in die nächste übernommen. Das erweist sich gerade bei der Eingabe von Stichworten und Kurznotizen in der Hektik des Arbeitstages als sinnvoll. Es versteht sich, daß man den gesamten Notizblock speichern kann.

Einen weiteren Block bilden Adreßdatei und Autodialer. Mit der Adreßdatei macht das Arbeiten richtig Spaß. Die Eingabemaske ist groß genug und verfügt über gute Editier- und Suchfunktionen. Ebenso verhält es sich mit dem Telefonmemo. Die Funktionen »Call completed« oder »Line busy«, – für deutsche Verhältnisse zwar sinnlos, da Sie nicht mit den Postmodems zusam-



»Cornerman« vereint eine ganze Menge nützlicher Funktionen auf engstem Raum



Eine davon ist das Plazieren der Fenster auf die gewünschte Bildschirmposition

menarbeiten – verschaffen eine gute Übersicht über die Telefongespräche. Eine einblendbare Uhr gilt schon fast als Standard für den Atari ST. Mit der »Cornerman«-Version hat es aber etwas Besonderes auf sich. Nicht nur die grafische Gestaltung – einer Analoguhr nachempfunden – fällt positiv auf. Ihre Nebenfunktion ist ungewöhnlich. Erscheint die Uhr nach dem Aktivieren noch als kleines Fenster auf dem Monitor, so wächst sie durch einen Mausklick auf die volle Bildschirmgröße. Mit der abschaltbaren Funktion, das Schrumpfen der Uhr nur nach Eingabe eines bis zu vier Zeichen langen und frei wählbaren Paßwortes zu erlauben, kann man wichtige Daten auf dem Bildschirm vor den Augen Neugieriger schützen. Das Einbrennen der Uhr in der Bildschirmmaske verhindert ein zyklischer Wechsel der Bildschirmfarbe. Eine außergewöhnliche Idee in dieser Runde der Hilfsprogramme für den ST ist auch das »Setup-Menü«. Alle wichtigen Parameter, die Cornerman nutzt, kann man hier vorgeben. So zum Beispiel die Anzeige einer winzigen Zeitausgabe in der rechten Ecke der Menüleiste und das Format dieser Anzeige. Ebenso ermöglicht es die RS232-, Taschenrechner-Parameter und das Paßwort der Analoguhr zu definieren und zu speichern. Durch den symbolisierten Bildschirm mit Funktionsfenstern legt man mit dem Mauszeiger fest, an welcher Stelle des Bildschirms sich die jeweiligen Fenster öffnen sollen.

Auch bei der Druckeransteuerung handelt es sich nicht um ein Menü, mit dem Steuerzeichen für Umlaute oder Schriftart an den Drucker gesendet werden. Vielmehr definiert man, welche Datenbereiche eines Unterprogramms auszudrucken sind. Als letzten Menü-Punkt enthält Cornerman »DOS WINDOW«. Befindet sich »COMMAND.TTP« aus dem Entwicklungspaket auf der Programmdiskette, erfolgt der Start nach dem Anklicken von »DOS WINDOW«. Die Eingabe der TOS-Befehle geht dann in der guten alten Eintipp-Manier vor sich. »Cornerman« beeindruckt durch einige außergewöhnliche Ideen. Leider ist es stark auf den amerikanischen Markt zugeschnitten.

(Ulli Jordan/hb)

Programmname	Hersteller	Preis	Umfang	Info
Calendar	Michtron	*	50 KByte	Microdeal,England
MacroDesk	Blue Moon Software	99 Mark	40 KByte	Finke, Wuppertal
Powerpack	Laser Soft	*	33 KByte	PDS, Rijswijk, Holland
Side Click	RDS	198 Mark	100 KByte	RDS, Raunheim
Cornerman	Michtron	134 Mark	100 KByte	Finke, Wuppertal

* Der Preis stand bei Redaktionsschluß noch nicht fest



Amiga-Guckloch

Nach langem Warten endlich da: Ein Speicher- und Diskettenmonitor für den Amiga. Seine besonderen Merkmale sind leistungsfähige Funktionen und eine einfache, logische Bedienung. Sein Name: C-Monitor.

Gleich vorweg: Der C-Monitor zählt zur Zeit zum Besten, was in dieser Art für den Amiga zu bekommen ist! Neben den Standard-Funktionen eines Monitors, wie dem Anzeigen und Manipulieren von Speicherinhalten, beherrscht der C-Monitor auch noch eine Reihe außergewöhnlicher Befehle, die beispielsweise Disketten-Daten verändern oder den geschützten RAM-Speicher des Amiga auslesen.

Vor der Erläuterung der einzelnen Funktionen sei noch kurz angemerkt, daß der Monitor nicht mit Pull-Down-Menüs oder ähnlichen Intuition-Schmankerln arbeitet. Dies ist nicht unbedingt als Nachteil anzusehen, da sich gezeigt hat, daß ein Monitor über Pull-Down-Menüs wesentlich langsamer zu bedienen ist als über die Tastatur. Zumal viele die Grundstruktur eines Monitors wohl noch vom Commodore 64 oder anderen Computern her kennen.

Das Programm macht in den ersten Minuten den Eindruck eines altbewährten Speicher-Monitors. Beabsichtigt oder nicht, man hat so als Umsteiger auf keinen Fall Probleme, sich mit diesem Monitor zurechtzufinden. Wie der Name schon erahnen läßt, ist der C-Monitor in der Programmiersprache C geschrieben, was eine ausreichende Geschwindigkeit garantiert.

Durch die sehr ausführliche Help-Funktion lernt man die Bedienung des Monitors schnell. Insgesamt sind in der Verkaufsversion 33 Hauptfunktionen (!) mit bis zu drei untergeordneten Funktionen implementiert. Die Vertriebsfirma Diamond Software plant auch regelmäßige und kostengünstige Updates zu ihrem C-Monitor.

Die Grundfunktionen dienen zum Auslesen des Speichers in Hexadezimalzahlen oder ASCII-Zeichen, dem Disassemblieren von 68000-Assembler-Programmen, dem Füllen von Speicherbereichen oder dem Suchen nach bestimmten Werten und Texten. Zu den erweiterten Befehlen, die zum großen Teil Amiga-spezifisch sind, zählt zum Beispiel das Laden von Programmen in einem Stück. Dieser Punkt muß näher erläutert werden, da nicht allgemein bekannt ist, daß der Amiga Programme nicht zusammenhängend lädt. Vielmehr legt das Betriebssystem ein Programm beim Laden in einzelnen Stücken, sogenannten Segmenten, im Speicher ab und springt bei der Ausführung des Programmes von einem Segment zum anderen.

Der Grund für diese umständliche Speicherverwaltung liegt darin, daß der Amiga in der Grundkonfiguration nur 256 KByte freien Speicher besitzt. Von diesem Speicher gehen nun allein schon rund 100 KByte für die Workbench verloren. Aus diesem Grund muß der Amiga mit seinem Speicherplatz sehr bedacht umgehen. Wenn ein Programm geladen wird, sucht der Amiga keinen freien Bereich, der der Länge des Programms entspricht, sondern füllt den Speicher beginnend mit der ersten freien Speicherstelle auf. Wie gesagt, ziemlich umständlich, aber sehr sparsam.

Man kann das Programm mit dem C-Monitor auch in einzelne Segmente unterteilt laden. Nur dann nämlich startet es auch der »g«-Befehl, so daß man es während des Editierens testen kann. Lädt man das Programm in einem Stück, gestaltet sich das Editieren einfacher und übersichtlicher, aber man muß auf das Starten des Programmes verzichten. Wurde das Programm segmentweise geladen, hilft beim Editieren und Disassemblieren ein Befehl, der die einzelnen Teile des Programms anzeigt. Damit erkennt man, wo welches Programm-Segment zu finden ist.

Auch das Speichern ganzer Speicherbereiche (inklusive des Betriebssystems) funktioniert mit dem C-Monitor bis auf eine Einschränkung zufriedenstellend. Man kann nur zusammenhängende Bereiche speichern. So ist es schwierig, ein Programm zu speichern, das sich in verschiedene Segmente untergliedert. Oder aber man speichert jedes Segment einzeln und fügt diese Files später mit dem JOIN-Kommando vom CLI aus zusammen.

Eine weitere Spezialität des C-Monitors stellen Befehle zum Einlesen von Spuren und Sektoren der Diskette (sogar des Kickstarts!) dar. Es ist auch möglich, den Bootblock, der Informationen über die Art der Diskette und deren Aufbau enthält, einzulesen, zu verändern und wieder zu speichern. So ist es theoretisch vorstellbar, ein eigenes Diskettenformat aufzubauen, da die Formatierungsroutine des Betriebssystems veränderbar ist. Es steht auch nichts im Wege, einen bestimmten Speicherbereich für den Amiga als »besetzt« zu definieren, so daß man nur noch selbst, vom Monitor aus, darauf zugreifen kann. Man lädt beispielsweise einen Sektor nach $30000, schützt diesen Bereich und eröffnet weitere Tasks, ohne daß diese die Sektordaten im Speicher überschreiben.

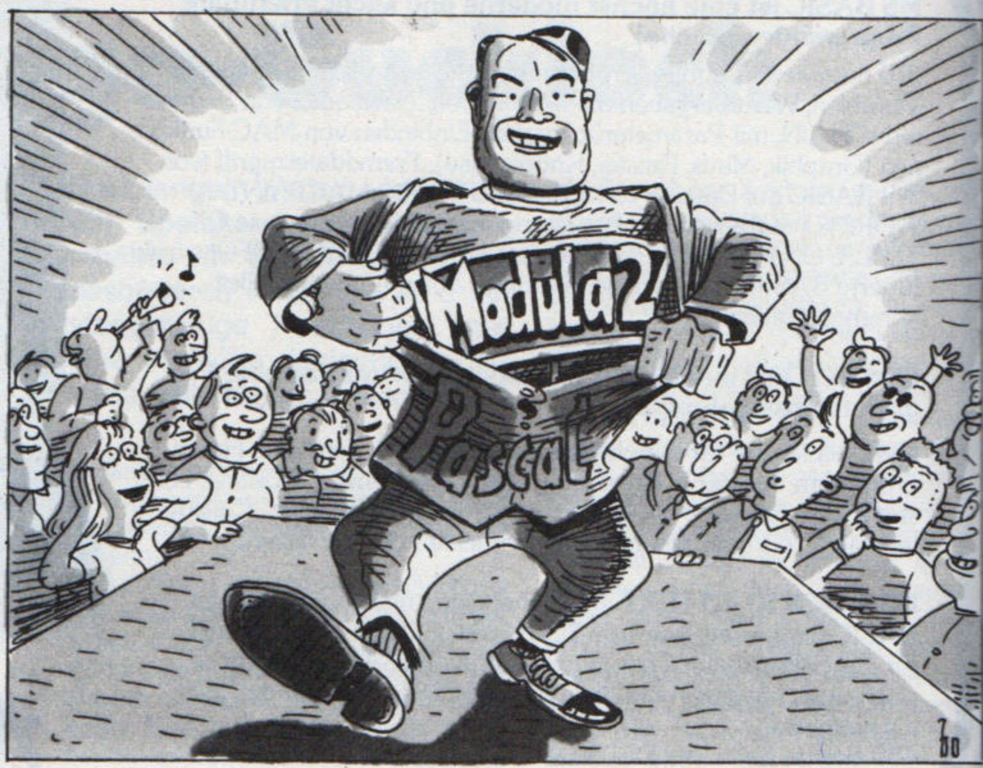
Zu den weiteren Befehlen im C-Monitor zählen das Umwandeln von dezimalen Zahlen in hexadezimale und umgekehrt, das Addieren, Subtrahieren, Dividieren und Multiplizieren mit Hexadezimal-Zahlen sowie das Verschieben und Vergleichen von Speicherbereichen. Alle Ausgaben können auch auf einem Drucker erfolgen. Das Ändern sämtlicher Register der 68000-CPU (auch des Supervisor-Stackpointers!) gehört zu den Befehlen, die zur Zeit kein anderer Monitor auf dem Amiga bietet.

Zudem ist ein leistungsfähiger Trace-Modus vorhanden, der es unter anderem erlaubt, Programme bis zu einem Break-Point ausführen oder in Einzelschritten ablaufen zu lassen. Leider arbeitet der Monitor nicht mit Assembler-Mnemonics, wobei man sich aber über den praktischen Nutzen eines integrierten Zeilen-Assemblers durchaus auch streiten kann.

Der Monitor ist sicherlich seinen nicht gerade niedrigen Preis wert, zumal auch noch günstige Updates angekündigt sind. Man darf gespannt sein, was das Haus Diamond Software noch zu bieten hat, wenn das erste Programm schon eine solch hohe Qualität aufweist. Wünschenswert wäre als Ergänzung zum C-Monitor ein ebenso einfach zu bedienendes und leistungfähiges Editor/Assembler-Paket, um in die Maschinensprach-Programmierung einzusteigen. (Ottmar Röhrig/ts)

Info: Diamond Software, Franz Hess, Sophienstr. 58, 7500 Karlsruhe. Preis: 139 Mark.





Von einem der auszog, Modula zu lernen...

Es gibt viele gute Gründe, in Modula zu programmieren. Steigen Sie ein in unseren großen Kurs, und gehen Sie auf die Reise durch eine moderne Programmiersprache.

Als im vergangenen Jahr der Atari ST auf dem Markt erschien, wurde schnell klar, daß das mitgelieferte ST-Basic kaum für sinnvolle Anwendungen einsetzbar ist. Die enorme Verschwendung von Speicherplatz, Ungenauigkeiten in den Rechenroutinen und eine zu gut gemeinte Fenstertechnik ließen bald die Freude am Programmieren dahinschwinden. Alle Pläne, mit dem ST schnelle und genaue Programme zu schreiben, fielen wie ein Kartenhaus zusammen. Zu jener Zeit befanden sich andere Basic-Interpreter und Pascal-Compiler lediglich im Ankündigungsstadium. Die Alternative, der in einem Entwicklungspaket enthaltene C-Compiler, erschien ungeeignet: Man hätte das gesamte Paket erstehen müssen. Guter Rat war also im wahrsten Sinn des Wortes teuer.

Daß ich dennoch die Not zur Tugend machen konnte, dafür sorgte wenig später ein Würfel. Er war Teil eines Demonstrationsprogrammes, in dem dreidimensionale Körper mit hoher Geschwindigkeit rotierten. Geschrieben war das Programm, Sie werden es erraten haben, natürlich in Modula. Angesichts dieser unglaublich flinken Grafikroutinen fielen die Würfel zugunsten des Modula2-Compilers von TDI.

Diese Version wird auf zwei einseitig randvoll beschriebenen Disketten geliefert. Dabei entfällt der Löwenanteil auf die Bibliotheksmodule, eine Sammlung von Unterprogrammen für den Benutzer, auf die wir später noch ausführlich eingehen werden.

Hardwareseitig kommt man einigermaßen mit einem 260 ST und einem einseitigen Laufwerk, also der Minimalkonfiguration, über die Runden. Jedoch wird mit dem häufigen Kopieren von der Compiler- auf die Linker-Diskette bald der Wunsch nach einem zweiten oder einem doppelseitigen Laufwerk wach. Für stolze Besitzer eines »Mega-ST« empfiehlt sich der Einsatz einer RAM-Disk. Hiermit ist dann Compilieren und Linken nur noch eine Sache von wenigen Augenblicken.

Was ist nun das Ziel dieses Kurses? In erster Linie soll derjenige, der noch keine Erfahrungen im Umgang mit einer Sprache wie Pascal oder C besitzt, in die Lage versetzt werden, eigene Programme in Modula zu formulieren. Das betrifft besonders die vielen Basic-Programmierer, die eine Alternative suchen, aber auch den Anfänger, der sich mit dieser Sprache von vornherein einen guten Programmierstil aneignen will. Zu guter Letzt soll der Kurs auch eingefleischten Pascal- oder C-Programmierern einen weitreichenden Überblick über Modula verschaffen. Natürlich kann er kein einführendes Buch ersetzen, da einfach nicht jede Einzelheit in aller Ausführlichkeit zur Sprache kommen kann.

Modula – der Name wird hier stellvertretend für Modula2 benutzt – ist die Fortentwicklung der Programmiersprache Pascal. Sie wurde ebenso wie Pascal von dem Schweizer Professor Niklaus Wirth an der Eidgenössischen

Technischen Hochschule in Zürich entwickelt. Ihre Geburtsstunde fällt in das Jahr 1979. Modula ist der Versuch, sowohl computernahe Elemente als auch Elemente von Hochsprachen zu einer modernen Sprache zu vereinigen. Modula2 , eine Weiterentwicklung von Modula, steuert ferner das Datenkonzept des Moduls und die Multitaskingfähigkeit (die Ausführung verschiedener Programme gleichzeitig) zur Sprache bei. Modula eignet sich gerade für die Programmierung großer Programme, beziehungsweise zur Softwareherstellung im Teamwork ganz besonders. Das zu erstellende Programm wird dabei in einzelne, voneinander unabhängige Teile (Module) unterteilt, die getrennt voneinander programmiert und getestet werden. Die Zusammenarbeit dieses »Puzzles« regeln genau definierte Schnittstellen, was eine Beeinflussung der Module untereinander ausschließt.

Was ist Modula?

Modula ist eine Compilersprache, die Erstellung eines Programmes erfolgt in mehreren Schritten: Zuerst geben Sie mit einem Editor den Programmtext (den sogenannten Quelltext) ein und speichern ihn auf Diskette.

Dieser Editor findet sich auf der Programmdiskette, es ist Ihnen jedoch freigestellt, auch Textverarbeitungsprogramme für Ihre Quelltexte einzusetzen. Als einzige Voraussetzung müssen Sie dasselbe Textformat auf der Diskette erzeugen wie der Editor.

Anschließend rufen Sie den Compiler auf und übermitteln ihm den Namen des Quelltextes. Dieser übersetzt (compiliert) den Quelltext in die Maschinensprache (Objectcode) Ihres Computers und legt die übersetzte Datei auf Diskette ab. Erkennt der Compiler einen Fehler im Quelltext, zum Beispiel einen unbekannten Befehl, so bricht die Compilierung ab und es erscheint eine Fehlermeldung auf dem Bildschirm. In diesem Fall rufen Sie wieder den Editor auf und verbessern den Fehler im Quelltext. Anschließend starten Sie erneut den Compiler. Dieses Wechselspiel zwischen Editor und Compiler wiederholt sich, bis der Compiler im Quelltext keine formalen Fehler mehr findet.

Anschließend rufen Sie den Linker auf. Linker bedeutet übersetzt »Binder«. Dieses Programm verbindet (linkt) Ihre in Maschinensprache vorliegenden Programme und Unterprogramme zu einem einzigen lauffähigen Programm. Die relativen Adressen der einzelnen Module werden dabei in absolute Adressen innerhalb des Gesamtprogramms umgesetzt.

Zu guter Letzt schließt sich die Testphase des fertigen Programmes an. Hierbei sollten alle möglichen und unmöglichen Anforderungen an das Programm gestellt werden, damit der Benutzer später absolute Narrenfreiheit genießt und nicht ein Absturz des Programms eventuell eine aufwendige Arbeit zerstört.

Leistet das Programm nicht das, was Sie sich ursprünglich vorgestellt haben, so widmen Sie sich wieder Ihrem Quelltext. Da dies vor allem den Anfänger unter Umständen frustriert, ist es ratsam, sich vor dem Eingeben des Quelltextes genaue Gedanken über die Wahl einer geeigneten Formulierung des Problemes in Modula zu machen. Dies setzt natürlich die genaue Kenntnis der Programmiersprache voraus und damit wären wir schon beim Thema: Dem Aufbau eines Modula-Programmes.

Ein Modula-Programm setzt sich aus Schlüsselsymbolen (Befehle), Standardnamen und benutzerdefinierten Zeichenfolgen zusammen. Schlüsselsymbole sind dem Compiler bekannte Zeichenfolgen, die eine festgelegte Funktion besitzen und im Programm nicht für andere Zwecke benutzt werden dürfen. Um sie von den anderen Zeichenfolgen zu unterscheiden, werden sie groß geschrieben. Die Liste aller Schlüsselsymbole finden Sie in Tabelle 1.

Standardnamen dienen anderen Aufgaben. Im Interesse der Austauschbarkeit von Modula-Programmen zwischen verschiedenen Computern sollten Sie diese nicht als Variablennamen umdefinieren. Tabelle 2 zeigt die reservierten Standardnamen. Deren Bedeutung und Funktion werden später erläutert.

AND	LOOP
ARRAY	MOD
BEGIN	MODULE
BY	NOT
CASE	OF
CONST	OR
DEFINITION	POINTER
DIV	PROCEDURE
DO	QUALIFIED
ELSE	RECORD
ELSEIF	REPEAT
END	RETURN
EXIT	SET
EXPORT	THEN
FOR	TO
FROM	TYPE
IF	UNTIL
IMPLEMENTATION	VAR
IMPORT	WHILE
IN	WITH

Tabelle 1. Alle reservierten Schlüsselsymbole. . .

ABS	INTEGER
ADDRESS	LONGCARD
ADR	LONGINT
BITSET	LONGREAL
BOOLEAN	MAX
CAP	MIN
CARDINAL	NEW
CHAR	NEWPROCESS
CHR	NIL
DEC	ODD
DISPOSE	ORD
EXCL	PROC
FALSE	REAL
FLOAT	SIZE
HALT	TRANSFER
HIGH	TRUE
INC	TRUNC
INCL	TSIZE
	VAL

Tabelle 2. . . .und Standardnamen

Daneben existieren die folgenden Operatoren:
+, –, *, /, :=, &, ˜, =, #, < >, >, <, > =, < =,
und die Trennzeichen:
( ) [](* *) ^ , ; . :

Die Operatoren und Trennzeichen sind mit denen aus Pascal identisch.

Selbstdefinierte Namen dienen der eindeutigen Bezeichnung von Daten und Objekten. Die Namen unterliegen den folgenden Einschränkungen:

1. Das erste Zeichen muß immer ein Buchstabe sein.
2. Ein Leerzeichen beendet den Namen und darf daher nicht zur Trennung eines Namens aus mehreren Wörtern verwendet werden.
3. Ein Unterstrich ist in Modula kein gültiges Zeichen.
4. Ein Name darf keine Operatoren enthalten.

Erlaubte selbstdefinierte Namen sind zum Beispiel: fuenfXdrittel, ein Testname, testwert1. Nicht verwenden dürfen Sie hingegen Namen wie: 5xdrittel, ein Testname, noch ein_Name.

Wie Sie sehen, unterscheidet Modula zwischen Groß- und Kleinbuchstaben, so daß Sie in der Lage sind, einen Namen auch aus mehreren Worten zusammenzusetzen.

Gewöhnen Sie sich im Interesse der Dokumentation und der besseren Lesbarkeit von vornherein an, den Quelltext mit ausreichenden Kommentaren zu versehen. Kommentare umschließen paarweise Klammern und dürfen an beliebiger Stelle im Programm stehen:
(* zum Beispiel so *)

Der Compiler überliest sie einfach und sie dürfen deshalb auch zwischen

Symbolen stehen, zum Beispiel so:
x (* Wert1 *) AND y (* Wert2 *)

Bevor Sie einen eigenen Namen zur Manipulation von Daten benutzen, müssen Sie ihn definieren. Dies geschieht im sogenannten Vereinbarungsteil eines Modula-Programms. Doch dazu später mehr. Neben den oben genannten Regeln gibt es noch Vereinbarungen über die Schreibweise von Zahlen. Man unterscheidet in Modula zwischen ganzen und reellen Zahlen. Ganze Zahlen (Integers) werden in drei verschiedenen Darstellungen verwendet:

1. Dezimaldarstellung

Hierbei werden die Ziffern in bekannter Manier ohne Zwischenräume aneinandergefügt. Zu den zulässigen Zeichen zählen das Vorzeichen und die Ziffern von 0 bis 9. Zum Beispiel: 1234 -39239 +374.

2. Oktaldarstellung

Die Zahlen werden zur Basis Acht dargestellt, das heißt, zulässig sind die Ziffern von 0 bis 7. Das letzte Zeichen muß der Buchstabe B sein. Zum Beispiel: 10B (=dezimal 8), 23B (=dezimal 19).

3. Hexadezimaldarstellung

Die Zahlen werden zur Basis Sechzehn dargestellt. Dabei sind folgende Ziffern gültig: 0123456789ABCDEF. Das letzte Zeichen in einer hexadezimalen Ziffernfolge muß der Buchstabe H sein. Die Zahlen 10 bis 15 werden durch die Buchstaben A bis F dargestellt. Ist die erste Stelle ein Buchstabe, so muß der Ziffernfolge eine Null voranstehen. Zum Beispiel: 23H (=dezimal 35), 0FFH (=dezimal 255).

Ein kleines Demonstrationsprogramm zeigt Ihnen den Umgang mit den verschiedenen Darstellungsarten:

```
MODULE zahlendarstellung;
  (* Importliste *)
FROM InOut IMPORT WriteString,
WriteLn,Write,WriteCard;
FROM Terminal IMPORT Read;
  (* Variablenvereinbarung *)
VAR   c:CHAR;
  (* Anweisungsteil *)
BEGIN
WriteString('Dezimalzahl "43"=');
WriteCard(43,3); WriteLn;
WriteString('Oktalzahl "43B"=');
WriteCard(43B,3); WriteLn;
WriteString('Hexzahl "43H"=');
WriteCard(43H,3); WriteLn;
Read(c)
(* Wartet auf Tastendruck *)
END zahlendarstellung.
```

Fließkommazahlen (das ist die Menge der rellen Zahlen) gliedern sich in Nachkommastellen (Mantisse) und Größenordnung (Exponent). Der Exponent gibt die Zehnerpotenz an, mit der die Mantisse zu Multiplizieren ist. Er ließe sich ersetzen durch die Vorschrift »...Mal Zehn hoch...«. So hat die Fließkommazahl 1.5E3 den Wert 1.534* 10^3 = 1534. Der Exponent wird durch ein großes E eingeleitet und darf weggelassen werden. Beispiele für erlaubte Fließkommazahlen sind :
-1.234E-4   1.0E2   3.14159

Sämtliche Zeichen lassen sich über ihren ASCII-Code ansprechen, wobei der Code in Oktaldarstellung, gefolgt von einem C, angegeben wird. Hierzu ein kleines Beispiel: Für die Ausgabe des Buchstaben a schreiben Sie normalerweise die Programmzeile »Write('a')«. Statt dessen ist aber ebenso »Write(141C)« erlaubt. Einen dritten Weg stellt die Ausgabe mit der Standardfunktion CHR dar. Als Argument übergeben Sie dieser Funktion den ASCII-Code des gewünschten Zeichens. Hiervon macht das folgende Programm Gebrauch:

```
MODULE chartest;
  (* Importliste *)
FROM Terminal IMPORT Write,
WriteLn,Read;
  (* Variablenvereinbarung *)
VAR c:CHAR;
  (* Anweisungsteil *)
BEGIN
c:='#';
  (* Zeichen im Klartext *)
Write(c);
WriteLn;
c:=CHR(43B);  (* Indirekte
Write(c);     Verwendung über
WriteLn;      den Zeichencode *)
c:=43C;       (* Direkte
Write(c);     Verwendung über
WriteLn;      den Zeichencode *)
Read(c)
END chartest.
```

Zeichenketten (Strings) begrenzen entweder Hochkommas oder Anführungszeichenstriche. Erlaubt sind also Stringzuordnungen durch: 'Das ist ein Test' "Und noch ein Test".

Nachdem Sie einen Überblick über die Schreibvereinbarungen gewonnen haben, sehen wir uns den Aufbau eines Modula-Programmes an. Betrachten Sie dazu eingehend das folgende Programmgerüst. Es verdeutlicht den prinzipiellen Aufbau eines sogenannten Hauptmoduls in Modula2, das auch als Modul mit der höchsten Ebene bezeichnet wird:

```
MODULE Programmname;
(* Importliste externer
Module *)
IMPORT Modulname1;
(* Importliste externer
Objekte *)
FROM Modulname2 IMPORT
Bezeichner;
(* Interne Module *)
MODULE Modulname3;
  IMPORT
  EXPORT
  MODULE
  CONST
  VAR
  PROCEDURE
BEGIN
 Anweisungsfolge
END Modulname3;
(* Konstantenvereinbarung *)
CONST
(* Eigendefinierte Datentypen *)
TYPE
(* Variablenvereinbarung *)
VAR
(* Prozedur-/Funktions-   *)
(* vereinbarung           *)
PROCEDURE Prozedurname;
(* Lokale Vereinbarungen *)
  CONST
  TYPE
  VAR
  PROCEDURE
BEGIN
 Anweisungsfolge
END Prozedurname;
(* Hauptprogramm *)
BEGIN
 Anweisungsfolge
END Programmname.
```

Teile dieses Programmgerüstes entfallen je nach Bedarf. Gehen wir zum nächsten Beispiel:

```
MODULE ErstesProgramm;
   (* Importliste *)
FROM Terminal IMPORT WriteString;
   (* Vereinbarungsteil *)
TYPE
String=ARRAY[0..80] OF CHAR;
VAR
testtext:String;
   (* Anweisungsteil *)
BEGIN
testtext:='Hallo, Modula-2.';
WriteString(testtext)
END ErstesProgramm.
```

Das Programm gibt – wie nicht anders zu erwarten – den Testtext: »Hallo, Modula-2« auf dem Bildschirm aus. Sie erkennen die klare Untergliederung eines Modula-Programmes in Vereinbarungs- und Anweisungsteil.

Diese beiden Teile analysieren wir nun Schritt für Schritt. Ein Modula-Programm beginnt grundsätzlich mit dem Schlüsselsymbol MODULE und einem Programmnamen. Das Ende des Programmes bildet immer das Symbol

END, gefolgt vom selben Programmnamen und einem Punkt. Dieser Punkt teilt dem Compiler mit, daß das Programm an dieser Stelle endet. Deshalb darf auch an keiner anderen Stelle im Programm eine Anweisung mit einem Punkt aufhören. An erster Stelle im Programm steht die sogenannte Importliste, die festlegt, welche Objekte aus den Bibliotheksmodulen im Programm verwendet werden. Da Modula im Sprachkern keine Ein- und Ausgabeoperationen enthält, müssen diese aus den Bibliotheksmodulen importiert werden. Importieren von Objekten bedeutet also das Einfügen der Objekte in das Programm beim Compilieren und Linken, so daß diese im Programm Verwendung finden. Dies ist die erste Anwendung des sogenannten Modulkonzeptes. Im obigen Beispiel wird die Prozedur WriteString, die zur Ausgabe einer Zeichenkette dient, aus dem Bibliotheksmodul »Terminal« eingebunden. Damit weiß der Compiler, was er mit dem Namen WriteString machen soll, wenn dieser im Quelltext auftaucht. Da dieses Wort kein Schlüsselsymbol ist, würde er sonst eine Fehlermeldung für ein nicht definiertes Wort ausgeben. Als nächster Abschnitt des Programmes folgt der sogenannte Vereinbarungsteil. Er enthält alle im nachstehenden Anweisungsteil verwendeten Daten und Datenstrukturen und weist ihnen einen Namen zu. In diesem Teil werden Konstanten, Typen, Variablen, Prozeduren, Funktionen und Module definiert. So wird in unserem Programm der Datentyp String definiert und der Variablen »testtext« dieser Datentyp zugewiesen. Damit weiß der Compiler, welche Zuweisungen an die Variable »testtext« zulässig sind und überprüft es beim Compilieren.

So ist Modula aufgebaut

An den Vereinbarungsteil schließt sich der Anweisungsteil an. Darin werden die Operationen für die Daten festgelegt. Der Anweisungsteil beginnt immer mit dem Schlüsselsymbol BEGIN und erstreckt sich dann bis zum Programmende. In unserem Programmbeispiel wird der Variablen »testtext« eine Zeichenkette zugewiesen und auf dem Bildschirm ausgegeben. Ganz nebenbei haben wir bereits den Begriff des Datentyps und der Variablen eingeführt. Hierzu noch einige Anmerkungen:

Typen sind dem Compiler bekannte Objekte bestimmter Speichergröße und Struktur. Die zulässigen Manipulationen an diesen Objekten liegen fest, das heißt der Anwender kann sie nicht ohne weiteres verändern. So ist zum Beispiel die Teilung eines Buchstabens (CHAR) durch eine Zahl eine unzulässige Manipulation und führt zu einer Fehlermeldung.

Um nun mehrere Objekte gleichen Typs zu manipulieren, müssen Sie diese unterschiedlich benennen. Dies geschieht in Form einer Variablenvereinbarung. Hierbei werden den Datentypen verschiedene Namen zugewiesen. Der Compiler reserviert dann jedem Namen (jeder Variablen) den zugehörigen Speicherplatz. So lassen sich verschiedenen Namen gleichen Typs unterschiedliche Werte zuweisen. Der Wert eines Namens ist änderbar, daher stammt auch der Name Variable. Die Vereinbarung geschieht in folgender Form:

```
VAR
name1,name2:Datentyp1;
name3 :Datentyp2;
```

Die Trennung der einzelnen Definitionen und Zuweisungen übernimmt ein Semikolon. So ist es zum Beispiel auch gestattet, die Variablenvereinbarung nebeneinander zu schreiben. Dies ist jedoch bei umfangreichen Deklarationsteilen unübersichtlich und erschwert die Lesbarkeit des Programmes.

Machen Sie es sich zur Gewohnheit, logisch zusammenhängende Anweisungen durch Einrücken zu kennzeichnen. Oftmals hilft es gerade bei größeren Programmen, wenn man auf den ersten Blick feststellt, wo Anweisungen einen Block bilden und ob dieser korrekt abgeschlossen wurde.

Typenvielfalt

In Modula unterscheidet man zwischen den sogenannten einfachen oder unstrukturierten Datentypen und den strukturierten Datentypen. Strukturiert bedeutet in diesem Zusammenhang, daß sich ein Datentyp in weitere Teile untergliedert, die getrennt voneinander ansprechbar sind. Daneben un-

terscheidet man noch zwischen den vordeklarierten Datentypen, die dem Compiler ohne weitere Vereinbarungen bekannt sind, und den vom Programmierer definierten Datentypen.

BOOLEAN

Der Datentyp BOOLEAN stellt eine logische Aussage dar. Er ist entweder wahr (TRUE) oder falsch (FALSE). TRUE und FALSE sind Schlüsselwörter. Die Werte dieses Typs werden durch Vergleichs- und logische Operationen gewonnen. Folgende Operationen stehen zur Verfügung:

=	Gleichheit
<>,#	Ungleichheit
<	kleiner als
<=	kleiner-gleich als
>	größer als
>=	größer-gleich als
IN	Enthalten in Operationen:
AND,&	Logisches UND
OR	Logisches ODER
NOT	Logische Verneinung

Da der Datentyp BOOLEAN Aussagen macht, hat es sich eingebürgert, den zugehörigen Variablen Adjektive als Namen zu geben. Werden mehrere Vergleiche und Operationen in einem Ausdruck verwendet, so ist eine gewisse Reihenfolge in deren Bearbeitung zu beachten: Die höchste Priorität besitzt NOT, gefolgt von AND und OR. Am Schluß der Hierarchie stehen die logischen Vergleichsoperationen (>, <, >= etc.).

Sollten Sie sich einmal über die Prioritäten nicht im klaren sein, so läßt sich die gewünschte Reihenfolge der Ausführung mit Klammern erzwingen. Vergleichsoperationen lassen sogar die Verknüpfung zweier Boolescher Werte zu, wobei FALSE < TRUE definiert ist.

BITSET

Der Datentyp BITSET repräsentiert eine Menge positiver ganzer Zahlen im Bereich von 0 bis N-1. So ist im TDI-Modula N=16, das heißt, eine Variable diesen Typs darf Werte von 0 bis 15 annehmen. Die in einer Menge enthaltenen Zahlen bezeichnet man als Elemente. Um den Inhalt der Menge festzulegen, bedient man sich der in der Algebra üblichen Mengenschreibweise durch geschweifte Klammern:

ungerade Zahl= {1,3,5,7,9,11,13,15}
leere Menge= {}
ganze Menge= {0..15} oder auch
ganze Menge= {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}

Folgende Operatoren verknüpfen die Mengen miteinander:

a+b bildet die Vereinigungsmenge. Die neue Menge enthält alle Elemente aus den Teilmengen a und b.

a-b bildet die Differenzmenge. Es werden alle Elemente von a abgezogen, die auch in b enthalten sind.

a*b bildet die Schnittmenge. Die neue Menge enthält nur die Elemente, die in beiden Teilmengen gemeinsam enthalten ist.

a/b bildet die symmetrische Differenz. Sie ist das genaue Gegenteil der Schnittmenge und enthält genau die Elemente, die den Teilmengen nicht gemeinsam sind.

Um festzustellen, ob eine Zahl in einer Menge enthalten ist, bedienen Sie sich der Testfunktion IN. Sie liefert als Ergebnis einen Booleschen Wert. So ordnet der Term »Frage =3 IN a« der Variablen »Frage« den Wert TRUE zu, sofern die Zahl 3 in der Menge a enthalten ist. Beachten Sie, daß »Frage« im Programm vorher als Boolescher Datentyp deklariert werden muß. Ferner ist beim Umgang mit Mengen zu beachten, daß keine Mengen mit Variablen gebildet werden dürfen. Um die Elemente einer Menge zu beeinflussen, existieren die beiden Standardprozeduren INCL und EXCL. Sie dienen dem Einfügen und Löschen von Elementen einer Menge. Sie erhalten als Argumente die Menge und das Element, das verändert werden soll. Die Anweisung INCL(a,7) fügt das Element mit dem Wert 7 in die Menge a ein, wohingegen EXCL(a,b) mit b=7 das Element 7 aus der Menge a entfernt. Das nachstehende Beispielprogramm demonstriert die Funktion der Mengenoperatoren:

```
MODULE bitsettest;
  (* Importliste *)
FROM TextIO IMPORT Write,
WriteString, WriteLn, WriteCard,
Read;
(* Variablendeklaration *)
VAR
a:BITSET;
c:CHAR;
  (* Prozedurdeklaration *)
PROCEDURE contents(x:BITSET);
(* Überprüft die Menge x auf  *)
(* die Elemente von 0 bis 15. *)
(* Falls TRUE dann Ausgabe.   *)
VAR i:CARDINAL;
BEGIN
FOR i:=0 TO 15 DO
  IF i IN x
   THEN WriteCard(i,2)
  END (* IF *)
END; (* FOR i *)
WriteLn
END contents;
BEGIN
WriteString('{1,2}+{6,7}:');
a:={1,2}+{6,7};
contents(a);
WriteString('{1..7}-{6..9}:');
a:={1..7}-{6..9};
contents(a);
WriteString('{3,5,7}*{6..9}:');
a:={3,5,7}*{6..9};
contents(a);
WriteString('{3,5,7}/{6..9}:');
a:={3,5,7}/{6..9};
contents(a);
Read(c)
END bitsettest.
```

Das Programm liefert die folgende Bildschirmausgabe:

{1,2}+{6,7}: 1 2 6 7
{1..7}-{6..9}: 1 2 3 4 5
{3,5,7}*{6..9}: 7
{3,5,7}/{6..9}: 3 5 6 8 9

CHAR

Der Datentyp CHAR dient der Behandlung einzelner Zeichen aus dem Zeichensatz des Computers. Sollen einzelne Zeichen den als CHAR deklarierten Variablen zugeordnet werden, so sind sie in Hochkommas oder Anführungszeichen zu setzen. Die Zuordnung erfolgt hier einfach mit dem Gleichheitszeichen. Steuerzeichen dagegen werden über ihre Ordnungszahl (ASCII-Wert) angesprochen. Dies geschieht, wie bereits erwähnt, durch Angabe des Zeichencodes in Oktaldarstellung. Ist der Code eines Zeichens nicht bekannt, so wird er mit der Standardfunktion ORD ermittelt. Das Ergebnis liefert eine positive Zahl, die sich anschließend im Programm weiterverarbeiten läßt. Das folgende kurze Programm liest von der Tastatur ein Zeichen ein und gibt den Code der gedrückten Taste aus, bis eine Null eingegeben wird:

```
MODULE Codetest;
FROM TextIO IMPORT WriteCard,
 WriteLn, WriteString, Read;
VAR a:CHAR;

BEGIN
REPEAT
 WriteString('Zeichen');
 Read(a);
 WriteString('besitzt den Code');
 WriteCard(ORD(a),3);
 WriteString(' dezimal.');
 WriteLn
UNTIL a='0'
END Codetest.
```

Wie der Zeichensatz Ihres Computers aufgebaut ist, entnehmen Sie bitte Ihrem Atari-Handbuch.

CARDINAL

Der Datentyp CARDINAL umfaßt die positiven ganzen Zahlen im Bereich von 0 bis 65535. Er benutzt also zur internen Darstellung 16 Bit für jede Zahl die-

ses Typs. Folgende Operationen sind auf CARDINAL anwendbar:

+	Addition
–	Subtraktion
*	Multiplikation
MOD	Modulo-Berechnung (Divisionsrest)
DIV	Ganzzahlige Division

Die Ganzzahldivision DIV schneidet die bei der Division auftretenden Nachkommastellen ab. Den Divisionsrest erhalten Sie über die Modulo-Funktion. Hierzu zwei Beispiele: 7 DIV 3 ergibt 2, 7 MOD 3 ergibt 1.

Einige Compiler stellen auch den Datentyp LONGCARD zur Verfügung, der die Zahlen von 0 bis zirka vier Milliarden umfaßt. In diesem Fall werden für jede Variable 32 Bit reserviert. Auf diesen Typ sind die gleichen Operationen anwendbar, wie auf seinen kleineren Bruder.

INTEGER

Dieser Datentyp umfaßt den Bereich der ganzen Zahlen von -32768 bis +32767. Auf ihn sind die gleichen Operationen wie beim Typ CARDINAL anwendbar, mit der Einschränkung, daß die Argumente bei der Modulo-Operation nicht negativ sein dürfen.

Ebenso wie bei CARDINAL existiert hier auch ein Datentyp LONGINT, der den Bereich +/- zwei Milliarden umfaßt.

REAL

Daten vom Typ REAL sind Zahlen in Fließkommadarstellung. Sie finden vor allem zur Berechnung von numerischen Werten in Mathematik und Technik Verwendung, da sie eine Teilmenge der reellen Zahlen darstellen, die nur durch die Anzahl der Nachkommastellen begrenzt ist.

Als Operationen sind die vier Grundrechnungsarten sowie Vergleichsoperationen zugelassen. Die Division erfolgt hier über den Operator »/« und nicht über die Standardfunktion DIV. Aufgrund der begrenzten Zahl an Nachkommastellen treten bei der Berechnung mit REAL unweigerlich Rundungsfehler auf. Bei Überprüfung auf Gleichheit zweier Werte ist deshalb darauf zu achten, daß der Vergleich besser als eine »Fast-Gleichheit« definiert wird, indem der Dezimalbereich, in dem Rundungen auftreten, nicht berücksichtigt wird. In Einsatzgebieten, die absolute Genauigkeit fordern, greift man deshalb soweit vertretbar auf Integer-Variablen zurück. Das folgende Programm demonstriert die Berechnung der Eulerschen Zahl e=2.718281824:

```
MODULE Euler;
(* Berechnet die Zahl e nach der
Eulerschen Formel *)
(* Importliste *)
FROM TextIO IMPORT Read,
WriteReal, WriteString;
(* Variablendeklaration *)
VAR
   e,ealt:REAL;
   i:CARDINAL;
   c:CHAR;
(* Prozedurdeklaration *)
PROCEDURE fak(x:CARDINAL):
CARDINAL;
(* Berechnet die Fakultät x! *)
         BEGIN
         IF x=0
          THEN RETURN 1
          ELSE RETURN x*fak(x-1)
         END (* IF *)
         END fak;
BEGIN
ealt:=0.0; (* alter Wert *)
e:=0.0;    (* aktueller Wert *)
i:=0;
REPEAT
 ealt:=e;   (* alt. Wert retten *)
 e:=e+1.0/FLOAT(fak(i));
 i:=i+1     (* Zähler erhöhen *)
UNTIL ABS(ealt-e)<1.0E-4;
            (* Abbruch? *)
(* Abbruch, wenn e auf vier
Stellen genau berechnet wurde *)
WriteString('Die Zahl e :');
WriteReal(e,7,4); (* Ausgabe *)
Read(c)     (* Tastendruck *)
END Euler.
```

Wenn Sie das Programm starten, stellen Sie fest, daß die Berechnung des gesuchten Wertes ziemlich ungenau ausfällt. Dies ist eine Folge der sich summierenden Rundungsfehler. Eine gewisse Verbesserung erreichen Sie unter Verwendung des genaueren Datentyps LONGREAL, den einige Compiler zur Verfügung stellen. LONGREAL bietet die doppelte Genauigkeit der Nachkommastellen gegenüber dem Datentyp REAL.

Genauere Informationen enthält das Handbuch zum TDI-Modula.

Sie kennen nun alle in Modula vordeklarierten Datentypen. Wenden wir uns nun einem weitaus interessanteren Teil zu, nämlich den Datentypen, auf deren Struktur der Programmierer Einfluß hat.

Datentypen selbstgestrickt

Modula gestattet, aus den vordefinierten Datentypen eigene Datentypen zu definieren. Dies geschieht im Vereinbarungsteil des Programms durch das Schlüsselwort TYPE. Sie legen neue Datentypen fest, indem Sie zum Beispiel schreiben:

```
TYPE
name1=CHAR;
name2=REAL;
name3=LONGCARD;
```

Im nachfolgenden Programm wird dann der Typ name1 genauso behandelt, wie der Typ CHAR. Natürlich hätte eine Umbenennung der vordefinierten Typen allein kaum einen programmtechnischen Nährwert. Aber mit diesem Sprachelement lassen sich aus den bisher besprochenen Datentypen CHAR, CARDINAL und INTEGER die sogenannten Unterbereichstypen vereinbaren.

Ein Unterbereich schränkt die zulässigen Werte eines Datentyps ein. Die Definition erfolgt durch Angabe der unteren und der oberen Grenze des zugelassenen Wertebereiches in eckigen Klammern. Die Grenzen werden im Vereinbarungsteil durch Konstante fixiert, indem Sie zum Beispiel schreiben:

```
CONST
 Grenze1=3;
 Grenze2=7*5+2;
TYPE
 Utyp=[Grenze1..Grenze2];
VAR
 Testvariable:Utyp;
```

Auch in der Variablenvereinbarung lassen sich die Bereichsgrenzen angeben:

```
VAR
 Testvariable:[3..37];
```

Die Verwendung der Konstanten- und Typenvereinbarung macht es aber einfacher, die Bereichsgrenzen zu verändern. Die Grenzen müssen konstante Ausdrücke und Grenze1 muß kleiner sein als Grenze2. Alle auf den Grundtyp zugelassenen Operationen sind auch auf seinen Unterbereichstyp anwendbar. Wird die untere Grenze als positiver Wert angegeben, so sind auf den neuen Typ alle Operationen von CARDINAL zugelassen, andernfalls die Operationen von INTEGER.

Aufzählungstypen

Wie der Name schon sagt, werden hier die erlaubten Werte des Datentyps durch Namensgebung aufgezählt. Hierzu einige Beispiele:

```
TYPE
 Maßeinheit=(kg,Liter,m,Joule);
 Hauptstadt=(Bonn,Wien,Paris);
 Hersteller=(Atari,Commodore,
             Apple, IBM,Andere);
```

Die Reihenfolge der Aufzählung legt die Ordnungszahl der Elemente fest. So hat im obigen Beispiel das Element kg die Ordnungszahl Null, Liter die Ordnungszahl Eins etc. Anhand dieser Ordnungszahl lassen sich auch Vergleiche

der Elemente einfach durchführen. Der Vergleich Paris > Wien liefert im obigen Beispiel den Wert TRUE. Ist die Ordnungszahl eines Elementes unbekannt, so ermitteln Sie diese durch die Standardfunktion ORD, indem Sie ihr das Element als Argument übergeben:
WieGroß: = ORD(Apple);

Mit den Aufzählungstypen erstellen Sie leicht lesbare Programme. Es ist sicherlich für jedermann einfacher verständlich, was zum Beispiel mit erkläre(Liter) als mit erkläre(1) gemeint ist.

Array- oder Feldtypen

Felder sind eine Aneinanderreihung verschiedener Elemente eines Datentyps. Sie stellen sich dies am besten anhand einer Rechnung vor, bei der zu jedem Posten ein Preis gehört. Auf die einzelnen Elemente des Feldes wird über einen sogenannten Index zugegriffen. Der Index steht in eckigen Klammern und bezeichnet die Platznummer eines Feldelements. Im Beispiel der Rechnung entspricht dieser der laufenden Nummer. Einen Feldtyp leitet immer das Schlüsselwort ARRAY ein. Danach erfolgt die Angabe der Dimension des Feldes, gefolgt vom Schlüsselwort OF und dem gemeinsamen Datentyp. Folgender Programmausschnitt zeigt die Deklaration von Feldvariablen:

```
TYPE
 linear=ARRAY[0..3] OF CHAR;
 quadratisch=ARRAY[0..3],[0..3]
 OF REAL;
 WieOft=ARRAY['A'..'Z'] OF
 CARDINAL;
VAR
 Hitliste:ARRAY[1..10] OF Lied;
```

Um nun beispielsweise auf den aktuellen Hit der Woche zuzugreifen, schreiben Sie später im Programm:

```
Sieger:=Hitliste[1];
```

Die Angabe der Dimension des Feldes erfolgt also wie die Definition eines Unterbereichstyps. Der Index wird als Unterbereichs-, Aufzählungstyp, CHAR oder als Boolescher Wert festgelegt. Als Operation mit den Feldelementen ist nur die Zuweisung erlaubt, Sie müssen also andere Operationen selbst in Unterprogrammen anlegen. Wie dies funktioniert, werde ich später noch zeigen. Eine der Anwendungen des Feldes ist die Programmierung von sogenannten Zeichenketten (Strings). Wie Sie von der Programmiersprache Basic her vielleicht wissen, besitzen Strings eine variable Länge. Dies ist in Modula nicht realisierbar. Man behilft sich, indem man Strings als Felder mit ausreichender Länge definiert. Das sieht dann zum Beispiel so aus:

```
TYPE
 String80=ARRAY[0..79] OF CHAR;
```

Die Angabe von Null als untere Grenze führt dazu, daß bei der Zuweisung kleinerer Zeichenketten die Variable mit dem ASCII-Codezeichen Null abgeschlossen wird. Ist die untere Grenze ungleich Null, so dürfen der Variablen nur elementeweise Zeichen zugewiesen werden, etwa durch folgende Anweisungen:

```
(* Elementeweise Zuweisung.      *)
(* Die FOR-Anweisung durchläuft
alle Werte von Null bis Länge,
mit der Schrittweite Eins        *)
     FOR i:=0 TO Länge DO
       atext[i]:=btext[i]
     END;
```

Diese Beispiel wollen wir nun in einem Programm anwenden:

```
MODULE stest;
  (* Importliste *)
FROM TextIO IMPORT Write,WriteLn,
WriteCard;
FROM Terminal IMPORT Read;
  (* Typenvereinbarung *)
TYPE
a=ARRAY[0..9] OF CHAR;
b=ARRAY[1..10] OF CHAR;
  (* Variablenvereinbarung *)
VAR
atext:a;
btext:b;
c:CHAR;
i:CARDINAL;
  (* Anweisungsteil *)
BEGIN
atext:='123';
(* kleiner als Variablenlänge  *)
(* Ausg. der Zeichen und Codes *)
FOR i:=0 TO 9 DO
    Write(atext[i]);
    WriteCard(ORD(atext[i]),4);
    WriteLn
END;
WriteLn;
(* Stringlänge=Variablenlänge  *)
atext:='1234567890123';
FOR i:=0 TO 9 DO
    Write(atext[i]);
    WriteCard(ORD(atext[i]),4);
    WriteLn
END;
Read(c)
END stest.
```

Der Nachteil des Feldtyps ist, daß alle Elemente vom gleichen Typ sein müssen. Um diese Klippe zu umschiffen, gestattet Modula die Programmierung von sogenannten Record- oder Verbundtypen.

Verbundtypen

Gehören zu einem Datensatz mehrere Objekte verschiedenen Typs, so lassen sich diese zu einem einzigen Datentyp zusammenfassen.

Nehmen wir hierzu wieder das Beispiel einer Rechnung, in der jedem Artikel Menge, Einzelpreis, Mehrwertsteuersatz und Gesamtpreis zugeordnet werden.
In Modula schreibt man hierfür:

```
TYPE
  Posten=RECORD
   Artikel:ARRAY[0..19] OF CHAR;
   Menge:CARDINAL;
   Einzelpreis:REAL;
   Mehrwertsteuersatz:CARDINAL;
   Gesamtpreis:REAL
  END;
VAR
  Rechnung:ARRAY[1..10] OF Posten;
  NeuEintrag:Posten;
```

Um nun auf die einzelnen Elemente der Rechnung zugreifen zu können, kommt die sogenannte »qualifizierende Namensgebung« zur Anwendung. Die Komponenten werden bei dieser Methode einzeln angesprochen und durch einen Punkt getrennt:
Verbundname.Komponentenname

Der entsprechende Programmteil für Neueinträge in die Rechnung sieht also etwa so aus:

```
NeuEintrag.Artikel:='Bleistift';
NeuEintrag.Menge:=3;
NeuEintrag.Einzelpreis:=0.40;
NeuEintrag.Mehrwertsteuer:=14;
NeuEintrag.Gesamtpreis:=1.20;
Rechnung[1]:=NeuEintrag;
```

Um nicht jedesmal den Verbundnamen mit angeben zu müssen, existiert die WITH-Anweisung. Sie bildet einen Block um Anweisungen, die sich jeweils auf denselben Verbund beziehen. Der Satzbau (die Syntax) sieht wie folgt aus:

```
WITH Verbundname DO
    Anweisungen
DO
```

Der Eintrag eines Postens in eine Rechnung hat damit die Form:

```
WITH NeuEintrag DO
   Artikel:='Bleistift';
   Menge:=3;
   Einzelpreis:=0.40;
   Mehrwertsteuersatz:=14;
   Gesamtpreis:=Menge*Einzelpreis
END
Rechnung[1]:=NeuEintrag;
```

Als einzige Operation ist wie auch beim Array nur die Zuweisung erlaubt. Allerdings ist es zulässig, die Variablen eines Verbundtyps beliebig miteinander zu verknüpfen. Mit dem Verbundtyp steht ein sehr flexibles Instrument zur Programmierung eigener Datenstrukturen zur Verfügung, da Ihnen bei der Auswahl der einzelnen Komponenten keinerlei Beschränkungen auferlegt werden. So ist es durchaus zulässig, als Komponententyp wiederum einen Ver-



bund zu wählen, so daß sich beliebige Verschachtelungen erreichen lassen. Existiert zum Beispiel ein Verbund folgender Gliederung:

```
TYPE
  name1=RECORD
          eins:CHAR;
          zwei:REAL
        END;
  name2=RECORD
          Element1:name1;
          Element2:CHAR
        END;
VAR
  test:name2;
```

So sprechen Sie die Komponente »eins« mit der Programmzeile:

```
test.Element1.eins:='a';
```

an.

Eine der wichtigsten Anwendungen stellt die Programmierung von Listen und Bäumen dar, in denen Verbunde durch Verweise (Zeiger) untereinander verknüpft werden. Um die Flexibilität der Verbunde noch weiter zu erhöhen, ist es sogar gestattet, die Komponenten in Abhängigkeit eines Unterscheidungselementes (Diskriminator) auszutauschen. Das Programm »varianttest« (Listing 1) demonstriert dies an einer einfachen Textausgabe. Es liest zwei Zahlen ein, deren eine als Inhalt des Verbundes und die zweite Zahl als Diskriminator fungiert. In Abhängigkeit dieser Zahl ist die variable Komponente entweder vom Typ CARDINAL oder vom Typ INTEGER. Anschließend gibt das Programm die feste Komponente als Zahl und die variable Komponente als Text aus.

Mengentypen

In Ergänzung zum Datentyp BITSET definieren Sie auch eigene Mengen mit Angabe des zulässigen Wertebereiches auf einfache Weise. Der Wertebereich der Menge muß ein Aufzählungs- oder Unterbereichstyp sein. Seine maximale Größe beschränkt sich auf 16 Elemente. Es sind alle Operationen wie beim Datentyp BITSET zugelassen, bis auf den Unterschied, daß bei der Zuweisung der Typname der Menge den geschweiften Klammern vorangestellt werden muß. Ein Beispiel:

```
TYPE
  Zeichen=SET OF ['A'..'C'];
VAR
  test:Zeichen;
```

Um nun der Variablen »test« ein Element zuzuweisen, müssen Sie die folgende Programmzeile hinzufügen:

```
test:=Zeichen{'A'};
```

Da der Wertebereich auf 16 Elemente beschränkt ist, erscheint der Anwendungsbereich ziemlich klein zu sein. In [2] (siehe Literaturverweise am Schluß) finden Sie jedoch ein Programm, daß diese Unzulänglichkeit behebt.

Die bis hierher besprochenen Datentypen haben den Nachteil, daß die Anzahl der Variablen im Programm feststeht und nur durch erneutes Compilieren änderbar ist. Es gibt jedoch viele Anwendungsgebiete, bei denen sich die Anzahl der zu verarbeitenden Daten während des Programmlaufs ändert oder deren Umfang beim Programmstart noch gar nicht bekannt ist. Zur Lösung dieses Problems bietet Modula die schon beim Verbundtyp erwähnten

Zeigertypen (Pointer)

Zeiger sind ganz allgemein Hinweise auf Speicherstellen, an denen sich Variablen befinden. Der Computer sieht in ihnen nichts anderes als Adressen. Aus der Sicht des Programmierers ist der Wert der Adresse aber nicht zugänglich und beeinflußbar. Dies ist weder notwendig noch möglich, da sich der Compiler um die Verwaltung des Speichers kümmert. Um aber dennoch direkt auf Variablen im Speicher zuzugreifen, bedient sich Modula der Zeiger in leicht abgeänderter Funktion:

Die Variable, auf die der Zeiger verweist, wird auch als Objekt bezeichnet. Das Objekt, auf das der Zeiger verweist, wird mit Hilfe eines an den Zeigernamen angefügten Hochpfeils angesprochen. Zeiger werden wie folgt definiert:

```
TYPE
  Zeigertyp=POINTER TO Objekttyp
VAR
  Zeiger:Zeigertyp;
```

Um nun dem Zeiger ein Objekt zuordnen zu können, muß ein freier Speicherplatz ausreichender Größe gesucht und dem Zeiger zugewiesen werden. Dies geschieht durch den Aufruf des Standardunterprogrammes NEW in der Form »NEW(Zeiger)«:

```
TYPE
  p=POINTER TO REAL;
VAR
  wert:p;
BEGIN
  ...
  NEW(wert);
  wert^:=1.234E-5;
  ...
END name.
```

Um den Speicherplatz für ein nicht mehr benötigtes Objekt wieder frei zu machen, rufen Sie einfach das Standardunterprogramm DISPOSE in der Form DISPOSE(wert) auf. Als Operationen auf Zeiger sind nur Vergleiche erlaubt. So besitzt zum Beispiel ein Zeiger, der noch keinen Speicherplatz besitzt, den Wert NIL (lies "Not in list"). Dies läßt sich durch die Anweisung »IF zeiger=NIL THEN..« einfach überprüfen.

Die Mächtigkeit des Datentyps POINTER wird erst in Zusammenhang mit dem Typ RECORD zugänglich, da gemeinsam mit diesem auch auf weitere Zeiger verwiesen wird und sich so die Daten dynamisch miteinander verknüpfen lassen. Enthält der Verbundtyp, auf den der Zeiger verweist, ebenfalls einen Zeiger, so nennt man diese Datenstruktur eine lineare Liste. Enthält der Verbundtyp mehrere Zeiger, so bezeichnet man diese Struktur als einen Baum. Diese beiden Datenstrukturen erlauben die Programmierung beliebig vernetzter Datenstrukturen.

Als Besonderheit gestattet Modula die Verwendung sogenannter Prozedurtypen, das heißt, Unterprogramme und Funktionen lassen sich als Variable definieren und verarbeiten.

Sie kennen nun alle Datentypen, die Modula bietet. Denken Sie immer daran: Die Festlegung von Datentypen dient der Zuordnung von Speichergröße und zulässigen Operationen. Um effizient in Modula zu programmieren, ist eine genaue Kenntnis aller Typen unabdingbar.

Im Laufe des Kurses haben wir mehrfach Zuweisungen benutzt, ohne deren Aufbau Beachtung zu schenken.

Zuweisungen dienen dazu, einer Variablen einen Wert zuzuordnen. Sie werden in der Form »Variable:=Ausdruck« vereinbart.

Von FOR bis CASE

Modula bietet jeden erdenklichen Komfort für die Programmierung von Schleifen und eine ganze Reihe weiterer Kontrollstrukturen. Die Befehle unterscheiden sich nur unwesentlich von den gleichnamigen Verwandten aus anderen Programmiersprachen. Sie sollen deshalb nur jeweils kurz erklärt werden:

Die REPEAT-Schleife hat die Form:

```
REPEAT
  Anweisungsfolge
UNTIL Boolescher Ausdruck
```

Die in der Schleife enthaltene Anweisungsfolge wird solange ausgeführt, bis der Boolesche Ausdruck wahr ist. Danach wird in der Programmzeile fortgefahren, die der UNTIL-Zeile unmittelbar folgt. Da die Anweisungsfolge mindestens einmal abgearbeitet wird, bezeichnet man diese Schleifenform auch als annehmende Schleifenform. Beispiel:

```
REPEAT        (* Liest Zeichen *)
```

```
Read(c)        (* von Tastatur, *)
UNTIL c='A' (* bis gleich A   *)
```

Im Gegensatz zur REPEAT-Schleife steht bei der WHILE-Schleife die Abbruchbedingung (Boolscher Ausdruck) am Anfang der Schleife. Sie wird auch als abweisende Schleife bezeichnet, da der erste Schleifendurchlauf nur stattfindet, sofern der Boolesche Ausdruck wahr ist. Sie hat folgende Form:

```
WHILE Boolescher Ausdruck
    Anweisungsfolge
 END
```

Oft ist es programmtechnisch erforderlich, den Abbruch eines Schleifendurchlaufs in der Schleife selbst zu programmieren. Die LOOP-Schleife erfüllt dieses Bedürfnis. Sie wird durchlaufen, bis das Programm innerhalb der Schleife den EXIT-Befehl erhält. Darauf setzt das Programm die Abarbeitung unmittelbar hinter END fort. LOOP-Schleifen haben die Form:

```
LOOP
  Anweisungsfolge( mit EXIT )
END
```

Ist bekannt, wie oft eine Anweisung ausgeführt werden muß, und wird eventuell der Zähler für die Schleifendurchläufe innerhalb der Anweisungsfolge benötigt, so findet die FOR-Schleife Verwendung. Sie hat die Form:

```
FOR count:=Wert1 TO Wert2 BY
Wert3 DO Anweisungsfolge
END
```

Die Anweisungsfolge wird solange ausgeführt, bis die Zählvariable count den Wert 2 erreicht hat. Count wird nach jedem Durchlauf um den Wert 3 erhöht. Die Zählvariable count kann vom Typ CARDINAL, INTEGER oder CHAR sein. Die Änderung von count erfolgt immer in ganzzahligen Schritten. Wird der Term »By Wert3« ausgelassen, so erhöht sich count bei jedem Durchlauf um 1. Das Hauptanwendungsgebiet in Modula liegt in der elementeweisen Verarbeitung von Daten des Feldtyps, wie bereits gezeigt wurde. Nach Durchlaufen der Schleife ist der Wert der Zählvariablen undefiniert.

Die IF-Anweisung dient dazu, in Abhängigkeit des Ergebnisses eines Booleschen Ausdrucks den Programmablauf in verschiedene Anweisungsfolgen verzweigen zu lassen. Der vollständige Aufbau der IF-Anweisung ist wie folgt (BA=Boolescher Ausdruck, AF=Anweisungsfolge):

```
IF BA THEN AF(1)
ELSIF BA THEN AF(2)
ELSIF BA THEN AF(3)
         ...
      ELSIF BA THEN AF(n)
      ELSE AF(n+1)
END
```

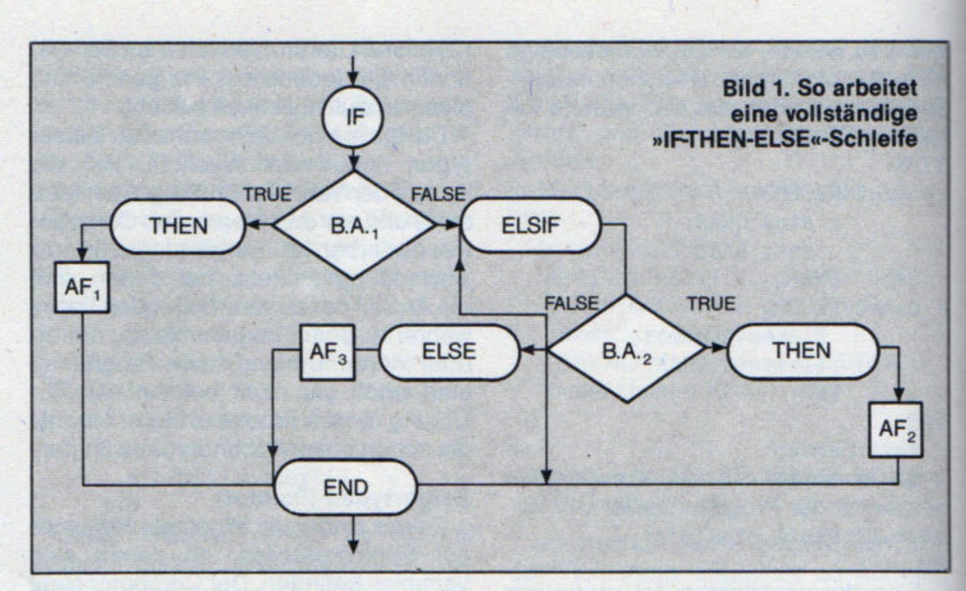


Bild 1. So arbeitet eine vollständige »IF-THEN-ELSE«-Schleife

Bei Erreichen der IF-Anweisung wird der erste Boolesche Ausdruck ausgewertet. Liefert er FALSE, so überspringt das Programm den Anweisungsteil hinter THEN und sucht hinter allen ELSIF und ELSE nach einem Booleschen Ausdruck, der TRUE liefert. Sobald ein TRUE entdeckt wird, führt das Programm den zugehörigen Anweisungsteil aus und verläßt anschließend die IF-Anweisung hinter der END-Zeile.

Abweichend von dem oben gezeigten Schema sind auch die folgenden Vereinfachungen zugelassen: »IF BA THEN AF(1) END« oder »IF BA THEN AF(1) ELSE AF(2) END«.

Das Flußdiagramm in Bild 1 stellt die Arbeitsweise der IF-Anweisung anschaulich dar.

In Listing 2, einem Programm, das die interne Uhr des Atari ST stellt, finden Sie ein Beispiel für eine IF-Anweisung.

Werden mehrere Fälle unterschieden, so ist die Anwendung der CASE-Anweisung übersichtlicher. Mit ihr überprüfen Sie Variablen auf verschiedene Werte, und lassen jedem von Ihnen bestimmten Wert einen Anweisungsteil folgen. Trifft keiner der angegebenen Werte zu, so läßt sich in einem ELSE-Teil eine Ausnahmehandlung programmieren. Die CASE-Anweisung hat die Form:

```
CASE Variable OF
  Wert1:Anweisungsfolge 1 |
  Wert2:Anweisungsfolge 2 |
     ...
  Wertn:Anweisungsfolge n
 ELSE Anweisungsfolge für Ausnah.
END
```

In Listing 4 finden Sie ein Beispiel für die CASE-Anweisung.

Oft muß in einem Programm eine Berechnung oder eine Operation mehrmals ausgeführt werden. Statt nun diesen Algorithmus so oft im Programm zu schreiben, wie er gebraucht wird, ist es einfacher und sinnvoller, ihn in Form eines Unterprogrammes zu programmieren. Man unterscheidet dabei Unterprogramme, denen entweder keine oder mehrere Parameter übergeben werden (Prozeduren) und solchen, die einen Wert zurückgeben (Funktionsprozeduren). Prozeduren und Funktionen müssen im Vereinbarungsteil deklariert werden und zwar in der Form:

```
PROCEDURE name(evtl. Parameter);
  lokale Vereinbarungen
BEGIN
       Anweisungsfolge
END name;
```

Betrachten wir also die verschiedenen Prozeduren der Reihe nach.

Parameterlose Prozeduren dienen lediglich dazu, einen selbständigen Anweisungsteil, der nicht auf Parameter anderer Programmteile zugreift, auszuführen. Dabei kann es sich um die Ausgabe einer Meldung handeln:

```
PROCEDURE gruß;
BEGIN
 WriteString('Hallo')
END gruß;
```

Der Aufruf der Prozedur findet einfach durch Angabe des Prozedurnamens im Programm statt.

Parameterbehaftete Prozeduren

Der Nutzen der Prozeduren wäre gering, ließen sich nicht die Berechnungen innerhalb der Prozedur mit verschiedenen Startwerten beginnen.

Bei der Prozedurdeklaration wird festgelegt, welche Parameter welchen Typs der Prozedur übergeben werden müssen. Dabei unterscheidet man zwischen Wertparametern (Call-by-value) und Variablenparametern (Call-by-reference). Bei Wertparametern wird der übergebene Wert in die Prozedur transportiert. Wird er in der Prozedur verändert, so ist dies für das aufrufende



Programm nicht sichtbar. Man sagt auch, es wird eine Kopie des Wertes an die Prozedur übergeben. Dies bedeutet, daß man als Parameter sowohl Variablen als auch zu berechnende Ausdrücke übergeben darf.

Im Gegensatz dazu wird bei den Variablenparametern an die Prozedur die Adresse der Variablen übergeben. Dies führt dazu, daß Veränderungen der Variablen in der Prozedur auch dem aufrufenden Programm bekannt werden. Bei dieser Übergabeform sind, wie der Name schon sagt, nur Variable als Parameter zugelassen. Dies wird in der Parameterliste durch das vorangestellte Schlüsselwort VAR festgelegt. An der Übergabetechnik ist eine weitere positive Eigenschaft der Prozeduren erkennbar: Das Prinzip der Lokalität. Das bedeutet, daß alle in der Prozedur auftretenden extra definierten Daten nur der Prozedur sichtbar sind. Dies ist der Vorzug der besonderen Arbeitsweise von Prozeduren: Die lokal definierten Daten werden erst beim Aufruf der Prozedur erzeugt und nach dem Abschluß der Prozedur wieder aus dem Speicher gelöscht.

Eine Parameterliste hat das folgende Format:»(VAR varname:Typ; name:Typ);«

In dem Beispiel »PROCEDURE f(VAR x:REAL; a,b:CARDINAL);« muß der Parameter x als Variable übergeben werden und ist vom Typ REAL. Hingegen sind die Parameter a und b vom Typ CARDINAL und werden als Werte übergeben.

Bei der Übergabe von Feldern tritt unter Umständen das Problem unterschiedlicher Feldlängen auf. Modula bietet hier die Übergabe sogenannter offener Felder als Parameter.

Soll eine Prozedur einen Wert explizit zurückgeben, so geschieht dies durch Anfügen eines Doppelpunktes und des Ergebnistypnamens an die Parameterliste. Solche Prozeduren heißen Funktionsprozeduren oder Funktionen. Das Ergebnis wird durch das Schlüsselwort RETURN an das aufrufende Programm übergeben. Das Beispiel berechnet die dritte Potenz einer Fließkommazahl:

```
PROCEDURE cube(x:REAL):REAL;
BEGIN
  RETURN x*x*x
END cube;
```

In Listing 5 finden Sie eine Reihe weiterer Beispielfunktionen. Als Besonderheit lassen sich in Modula die Prozeduren wie Datentypen als sogenannte Prozedurtypen behandeln:

```
TYPE
  funktion=PROCEDURE(REAL):REAL;
VAR
  x:funktion;
```

Außerdem lassen sich Prozeduren als Parameter an andere Prozeduren übergeben. Beispiele hierzu finden Sie in der weiterführenden Literatur am Ende dieses Kurses.

An dieser Stelle schließen die Ausführungen über die Sprachelemente von Modula.

Modern mit Modulen

Module haben Sie bereits im Rahmen der bisher gezeigten Programme kennengelernt. Ein Modul stellt eine eigenständige Programmierumgebung dar, außerhalb der keine Details des Inhalts »sichtbar« sind. Ist ein Programm, Modul oder Variable sichtbar, so kann ein »sehendes« Programm hierauf zugreifen und es manipulieren. Diese Sichtbarkeitsgrenze kann der Programmierer gezielt durch Angabe sogenannter Import- und Exportlisten umgehen. Diese Listen standardisieren den Datenaustausch zwischen Modulen und gestatten das unabhängige Erstellen von Programmen durch Unterteilen der Aufgabenstellung.

Lediglich die Schnittstellen (die Import- und Exportlisten) müssen vorher vereinbart worden sein. Ebenso nützlich ist es, Algorithmen und Daten eines Hauptprogrammes in sogenannten inneren Modulen zu verpacken, da sich so die Beeinflussung mehrerer Module untereinander ausschließt. Schließlich lassen sich Daten und Prozeduren, die immer wieder benötigt werden, in sogenannten Bibliotheksmodulen niederlegen, so daß Sie sie nicht jedesmal wieder neu schreiben müssen. Diese werden unterteilt in einen sogenannten Definitionsmodul und einen Implementationsmodul. Beide Module werden getrennt vom Hauptprogramm übersetzt und als Bibliotheksmodul gespeichert. Beginnen wir beim inneren Modul. Ein inneres Modul wird im Vereinbarungsteil eines Hauptprogrammes wie folgt definiert:

```
MODULE name;
IMPORT name1;
FROM modulname2 IMPORT name2;
EXPORT name3;
EXPORT QUALIFIED name4;
(* Vereinbarungsteil *)
(* Anweisungsteil *)
END name;
```

Wie Sie sehen, unterscheidet sich ein inneres Modul nur durch die Angabe einer Exportliste von einem Hauptprogramm. Da auch ein inneres Modul einen Vereinbarungsteil besitzt, lassen sich innere Module beliebig tief ineinander verschachteln. Den Datenaustausch zwischen ihnen regeln die schon erwähnten Import- und Exportlisten.

Dabei sind folgende Regeln zu beachten:

- Wird ein Objekt exportiert, so ist es im übergeordneten Modul sichtbar, als wäre es dort definiert worden.
- Wird ein Objekt durch das Schlüsselwort QUALIFIED exportiert, so muß der Modulname zusammen mit einem nachgestellten Punkt dem Objektnamen vorangestellt werden. So lassen sich aus verschiedenen Modulen Objekte gleichen Namens importieren.
- Wird ein komplettes Modul importiert, so müssen die importierten Objekte durch Angabe des Modulnamens qualifiziert werden: »Modulname.Objektname«.
- Standardnamen und Standardprozeduren werden automatisch in alle Module importiert. Ein Bibliotheksmodul gliedert sich in in zwei Teile. Das Definitionsmodul enthält die Vereinbarung aller zu exportierenden Objekte und wird in der folgenden Form gebildet:

```
DEFINITION MODULE name;
(* Importlisten *)
(* Exportlisten *)
(* Vereinbarungen *)
END name.
```

Beim Import von Objekten aus Bibliotheksmodulen gelten dieselben Regeln wie bei den inneren Modulen. Zu jedem Definitionsmodul gehört das Implementationsmodul gleichen Namens. Im Aufbau ähnelt es dem Definitionsmodul:

```
IMPLEMENTATION MODULE name;
(* Import-/Exportlisten *)
(* Vereinbarungsteil *)
BEGIN
(* Anweisungsfolgen *)
END name.
```

Es gilt dabei folgende Regeln zu beachten:

- Im Definitionsmodul vereinbarte Typen und Variablen sind auch im Implementationsmodul bekannt und brauchen nicht noch einmal vereinbart werden.
- Der Anweisungsteil wird beim Import von Objekten aus dem Bibliotheksmodul zuerst ausgeführt.

Hiermit sind wir am Ende unseres Einführungskurses in Modula2 angekommen.

(Hermann Reißig/Matthias Rosin/lg)

Literatur

[1] N. Wirth, »Programmieren in Modula-2«, Springer-Verlag

[2] Dal Cin/Lutz/Risse, »Programmierung in Modula-2«, Teubner Studienskripte Nr. 100

[3] G. Pomberger, »Softwaretechnik und Modula-2«, Hanser-Verlag

[4] N. Wirth, »Algorithmen und Datenstrukturen«, Teubner-Verlag

```
MODULE varianttest; (* HR,15.04.86 *)
  (* Importliste *)
FROM Terminal IMPORT Read, Write, WriteString, WriteLn;
FROM InOut IMPORT ReadCard,WriteCard;
  (* Vereinbarungsteil *)
TYPE test=RECORD
          nummer:CARDINAL;
          CASE x:BOOLEAN OF              (* x waehlt aus, welcher Para-*)
               TRUE: para:CARDINAL | (* meter verwendet wird.      *)
               FALSE:zahl:INTEGER    (* x=Diskriminator(para,zahl) *)
          END (* CASE *)
        END; (* RECORD *)
VAR a  :test;
    c  :CHAR;
    d  :BOOLEAN;
  (* Anweisungsteil *)
BEGIN
REPEAT
  WriteString('*** Programm zum Testen von varianten Records ***');
  WriteLn;
  WriteString('Einlesen des varianten Records:'); WriteLn;
  WITH a DO
       WriteString('Nummer?');
       ReadCard(nummer);
       WriteLn;
       WriteString('Druecken von 1 weißt CARDINAL=1 zu,');
       WriteString(' alles andere fuehrt zu INTEGER=-1.');
       WriteLn;
       WriteString('Bitte Taste druecken...');
       Read(c);
       WriteLn;
       WriteString('Eingegebenes Zeichen: '); Write(c);
       WriteLn;
       x:=(c='1');
       CASE x OF
            TRUE :para:=1 |
            FALSE:zahl:=-1
       END (* CASE *)
  END; (* WITH *)
  WriteString('Abfragen des varianten Records:'); WriteLn;
  WITH a DO
       WriteString('Nummer=');
       WriteCard(nummer,2);
       WriteLn;
       CASE x OF
            TRUE :WriteString('CARDINAL: plus');
                  WriteCard(para,2);
                  WriteLn |
            FALSE:WriteString('INTEGER:  minus');
                  WriteCard(ABS(zahl),2);(* Typwandlung durch *)
                  WriteLn               (* Typanpassung      *)
       END; (* CASE *)
  WriteLn
  END; (* WITH *)
  WriteString('[ESC] -> Programmende'); WriteLn;
  Read(c);
  WriteLn
UNTIL c=CHR(27)
END varianttest.,
```

Listing 1. Varianttest

```
MODULE startuhr;
(* 1986, Hermann Reissig, Werner-Heisenberg-Weg 39/3C, *)
(*                8014 Neubiberg                        *)
(* Falls sich das Programm im AUTO-Ordner befindet, wird beim *)
(* Einschalten des Rechners die Uhrzeit eingelesen *)
FROM InOut IMPORT ReadCard,WriteString,WriteLn;
FROM GEMDOS IMPORT SetDate,SetTime;
FROM Terminal IMPORT Write,Read;
VAR
min,hour,day,month,year:CARDINAL;
date,time:CARDINAL;
c:CHAR;
BEGIN
Write(CHR(27)); Write('E');  (* Clear Home *)
WriteString('********************* S T A R T U H R.P R G  ');
WriteString('Ver. 2.0 *********************'); WriteLn;
WriteString(' (C) 1986, Hermann Reissig, Werner-Heisenberg-Weg 39/3C, ');
WriteString('8014 Neubiberg'); WriteLn;
WriteString('********************* S T A R T U H R.P R G  ');
WriteString('Ver. 2.0 *********************'); WriteLn; WriteLn;
WriteString('Uhr initialisieren ? <j/n>');
Read(c); WriteLn;
(* Abbruch des Programms, falls erwünscht *)
IF ((c='j') OR (c='J')) THEN
     WriteString('Bitte eingeben...'); WriteLn; WriteLn;
     (* Datum und Zeit werden in zwei 16-Bit-Worten abgelegt *)
     (* Die einzelnen Daten besitzen aufgrund der Zuordnung  *)
     (* zu einer Position im Wort eine bestimmte Wertigkeit  *)
     min:=0; (* Wertigkeit 32 *)
     hour:=0; (* Wertigkeit 2048 *)
     day:=0; (* Wertigkeit 1 *)
     month:=0; (* Wertigkeit 32 *)
     year:=0; (* Wertigkeit 512 *)
     WriteString('Jahr  <85...00>:');
     ReadCard(year); WriteLn;
     WriteString('Monat  <1...12>:');
     ReadCard(month); WriteLn;
     WriteString('Tag    <1...31>:');
     ReadCard(day); WriteLn;
     WriteString('Stunde <0...23>:');
     ReadCard(hour); WriteLn;
     WriteString('Minute <0...59>:');
     ReadCard(min);
     time:=32*min+2048*hour;
     date:=day+32*month+512*(year-80);
     SetDate(date); (* Bibliotheksfunktionen aus TDI-Modula zum *)
     SetTime(time)  (* Setzen von Uhrzeit und Datum             *)
ELSE WriteString('Programm abgebrochen.')
END (* IF *)
END startuhr.
```

Listing 2. Startuhr als Beispiel für die IF-Anweisung

```
DEFINITION MODULE Super;

PROCEDURE SuperIn;  (* Schaltet Supervisor-Modus ein; USP in A2 *)
PROCEDURE SuperOut; (* Schaltet Supervisor-Modus aus *)

END Super.

  (* LISTING 3b *)
IMPLEMENTATION MODULE Super;

FROM SYSTEM IMPORT CODE;

PROCEDURE SuperIn; (* EINSCHALTEN DES SUPERVISOR-MODUS *)
BEGIN
CODE(42A7H,3F3CH,0020H,4E41H,5C8FH,2440H)
(*
   Assembler-Quelltext:
42A7        clr.l    -(sp)
3F3C  0020  move.w   #$20,-(sp)
4E41        trap     #1
5C8F        addq.l   #6,sp
2440        move.l   d0,a2
*)
END SuperIn;

PROCEDURE SuperOut; (* AUSSCHALTEN DES SUPERVISOR-MODUS *)
BEGIN
CODE(2F0AH,3F3CH,0020H,4E41H,5C8FH);
(*
  Assembler-Quelltext:
2F0A        move.l    a2,-(sp)
3F3C  0020  move.w    #$20,-(sp)
4E41        trap      #1
5C8F        addq.l    #6,sp
*)
END SuperOut;
END Super.
```

Listing 3. Mit diesem Modul schalten Sie in den Supervisormodus

```
MODULE sysvar;
(*  Monitor der Systemvariablen der ATARI-ST-Computer  *)
FROM SYSTEM   IMPORT ADDRESS;
FROM TextIO   IMPORT WriteString,WriteLn;
FROM Terminal IMPORT Read,Write;
FROM Super    IMPORT SuperIn,SuperOut;
FROM Binary   IMPORT and;

VAR c:CHAR;

PROCEDURE WriteMem(z:ADDRESS; l:CHAR);
VAR first,second,y:CARDINAL;

        PROCEDURE WriteAsHexDigit(x:CARDINAL);
            BEGIN
                IF x<10 THEN Write(CHR(x+48)) (* 48=Zeichen "0" *)
                         ELSE Write(CHR(x+55)) (* 55=Zeichen "7" *)
                END
            END WriteAsHexDigit;

BEGIN
CASE l OF
'L': (* Longword *)
       (* Oberes Wort bearbeiten *)
       y:=and(CARDINAL(z^),0FF00h); (* Ausblenden des unteren Bytes *)
       y:=y DIV 256;  (* Um ein Wort nach rechts schieben *)
       first:=y DIV 16; (* Oberes Nibble *)
       second:=y MOD 16; (* Unteres Nibble *)
       WriteAsHexDigit(first);
       WriteAsHexDigit(second);
       y:=and(CARDINAL(z^),0FFh); (* Ausblenden des oberen Bytes *)
       first:=y DIV 16; (* Oberes Nibble *)
       second:=y MOD 16; (* Unteres Nibble *)
       WriteAsHexDigit(first);
```

```
        WriteAsHexDigit(second);
        (* Unteres Wort bearbeiten; sonst wie oben *)
        z:=z+2;
        y:=and(CARDINAL(z^),0FF00h);
        y:=y DIV 256;
        first:=y DIV 16;
        second:=y MOD 16;
        WriteAsHexDigit(first);
        WriteAsHexDigit(second);
        y:=and(CARDINAL(z^),0FFh);
        first:=y DIV 16;
        second:=y MOD 16;
        WriteAsHexDigit(first);
        WriteAsHexDigit(second)|
'W': (* Word *)
        y:=and(CARDINAL(z^),0FF00h);
        y:=y DIV 256;
        first:=y DIV 16;
        second:=y MOD 16;
        WriteString('    ');
        WriteAsHexDigit(first);
        WriteAsHexDigit(second);

        y:=and(CARDINAL(z^),0FFh);
        first:=y DIV 16;
        second:=y MOD 16;
        WriteAsHexDigit(first);
        WriteAsHexDigit(second)|
'B': (* Byte *)
        y:=and(CARDINAL(z^),0FFh);
        first:=y DIV 16;
        second:=y MOD 16;
        WriteAsHexDigit(first);
        WriteAsHexDigit(second)|
ELSE
        WriteString('*** nicht implementiert ! ***');
END;
END WriteMem;
BEGIN
WriteString('*********************** Systemvariablen **************');
WriteLn;
WriteString('**************** Alle Angaben in hexadezimal ***********');
WriteLn;
SuperIn; (* Supervisormodus einschalten, da sonst Busfehler ! *)
WriteString('etv_timer   :'); WriteMem(400h,'L'); WriteString(' | ');
WriteString('etv_critic  :'); WriteMem(404h,'L'); WriteString(' | ');
WriteString('etv_term    :'); WriteMem(408h,'L'); WriteLn;
WriteString('etv_xtra1   :'); WriteMem(40Ch,'L'); WriteString(' | ');
WriteString('etv_xtra2   :'); WriteMem(410h,'L'); WriteString(' | ');
WriteString('etv_xtra3   :'); WriteMem(414h,'L'); WriteLn;
WriteString('etv_xtra4   :'); WriteMem(418h,'L'); WriteString(' | ');
WriteString('etv_xtra5   :'); WriteMem(41Ch,'L'); WriteString(' | ');
WriteString('memvalid    :'); WriteMem(420h,'L'); WriteLn;
WriteString('memctrl     :'); WriteMem(424h,'W'); WriteString(' | ');
WriteString('resvalid    :'); WriteMem(426h,'L'); WriteString(' | ');
WriteString('resvector   :'); WriteMem(42Ah,'L'); WriteLn;
WriteString('phystop     :'); WriteMem(42Eh,'L'); WriteString(' | ');
WriteString('_membot     :'); WriteMem(432h,'L'); WriteString(' | ');
WriteString('_memtop     :'); WriteMem(436h,'L'); WriteLn;
WriteString('memval2     :'); WriteMem(43Ah,'L'); WriteString(' | ');
WriteString('flock       :'); WriteMem(43Eh,'W'); WriteString(' | ');
WriteString('seekrate    :'); WriteMem(440h,'W'); WriteLn;
WriteString('_timer_ms   :'); WriteMem(442h,'W'); WriteString(' | ');
WriteString('_fverify    :'); WriteMem(444h,'W'); WriteString(' | ');
WriteString('_bootdev    :'); WriteMem(446h,'W'); WriteLn;
WriteString('palmode     :'); WriteMem(448h,'W'); WriteString(' | ');
WriteString('defshiftmod :'); WriteMem(44Ah,'W'); WriteString(' | ');
WriteString('sshiftmod   :'); WriteMem(44Ch,'W'); WriteLn;
WriteString('_v_bas_ad   :'); WriteMem(44Eh,'L'); WriteString(' | ');
WriteString('vblsem      :'); WriteMem(452h,'W'); WriteString(' | ');
WriteString('nvbls       :'); WriteMem(454h,'W'); WriteLn;
WriteString('_vblqueue   :'); WriteMem(456h,'L'); WriteString(' | ');
WriteString('colorptr    :'); WriteMem(45Ah,'L'); WriteString(' | ');
WriteString('screenptr   :'); WriteMem(45Eh,'L'); WriteLn;
WriteString('_vbclock    :'); WriteMem(462h,'L'); WriteString(' | ');
WriteString('_frclock    :'); WriteMem(466h,'L'); WriteString(' | ');
WriteString('hdv_init    :'); WriteMem(46Ah,'L'); WriteLn;
WriteString('swv_vec     :'); WriteMem(46Eh,'L'); WriteString(' | ');
WriteString('hdv_bpb     :'); WriteMem(472h,'L'); WriteString(' | ');
WriteString('hdv_rw      :'); WriteMem(476h,'L'); WriteLn;
WriteString('hdv_boot    :'); WriteMem(47Ah,'L'); WriteString(' | ');
WriteString('hdv_mediach :'); WriteMem(47Eh,'L'); WriteString(' | ');
WriteString('_comload    :'); WriteMem(482h,'L'); WriteLn;
WriteString('conterm     :'); WriteMem(484h,'W'); WriteString(' | ');
WriteString('trp14ret    :'); WriteMem(486h,'L'); WriteString(' | ');
WriteString('criticret   :'); WriteMem(48Ah,'L'); WriteLn;
WriteString('themd0      :'); WriteMem(48Eh,'L'); WriteString(' | ');
WriteString('themd1      :'); WriteMem(492h,'L'); WriteString(' | ');
WriteString('themd2      :'); WriteMem(496h,'L'); WriteLn;
WriteString('themd3      :'); WriteMem(49Ah,'L'); WriteString(' | ');
WriteString('____md      :'); WriteMem(49Eh,'L'); WriteString(' | ');
WriteString('saveptr     :'); WriteMem(4A2h,'L'); WriteLn;
WriteString('_nflops     :'); WriteMem(4A6h,'W'); WriteString(' | ');
WriteString('con_state   :'); WriteMem(4A8h,'L'); WriteString(' | ');
WriteString('save_row    :'); WriteMem(4ACh,'W'); WriteLn;
WriteString('sav_context :'); WriteMem(4AEh,'L'); WriteString(' | ');
WriteString('_bufl0      :'); WriteMem(4B2h,'L'); WriteString(' | ');
WriteString('_bufl1      :'); WriteMem(4B6h,'L'); WriteLn;
WriteString('_hz_200     :'); WriteMem(4BAh,'L'); WriteString(' | ');
WriteString('the_env     :'); WriteMem(4BCh,'L'); WriteString(' | ');
WriteString('_drvbits    :'); WriteMem(4C2h,'L'); Wrb4eLn;
WriteString('_dskbufp    :'); WriteMem(4C6h,'L'); WriteString(' | ');
WriteString('_autopath   :'); WriteMem(4CAh,'L'); WriteString(' | ');
WriteString('_vbl_list0  :'); WriteMem(4CEh,'L'); WriteLn;
WriteString('_vbl_list1  :'); WriteMem(4D2h,'L'); WriteString(' | ');
WriteString('_vbl_list2  :'); WriteMem(4D6h,'L'); WriteString(' | ');
WriteString('_vbl_list3  :'); WriteMem(4DAh,'L'); WriteLn;
WriteString('_vbl_list4  :'); WriteMem(4DEh,'L'); WriteString(' | ');
WriteString('_vbl_list5  :'); WriteMem(4E2h,'L'); WriteString(' | ');
WriteString('_vbl_list6  :'); WriteMem(4E6h,'L'); WriteLn;
WriteString('_vbl_list7  :'); WriteMem(4EAh,'L'); WriteString(' | ');
WriteString('_dumpflg    :'); WriteMem(4EEh,'W'); WriteString(' | ');
WriteString('_ptrabt     :'); WriteMem(4F0h,'W'); WriteLn;
WriteString('_sysbase    :'); WriteMem(4F2h,'L'); WriteString(' | ');
WriteString('_shell_p    :'); WriteMem(4F6h,'L'); WriteString(' | ');
WriteString('end_os      :'); WriteMem(4FAh,'L'); WriteLn;
WriteString('exec_os     :'); WriteMem(4FEh,'L');
SuperOut;
Read(c);
WriteLn;
Read(c)
END sysvar.
```

Listing 4. Sysvar zeigt die Systemvariablen des ST

```
DEFINITION MODULE Binary; (* 30.04.86 *)
(* 1986 by Hermann Reissig, Werner-Heisenberg-Weg 39/3C, 8014 Neubiberg*)

PROCEDURE and(x,y:CARDINAL):CARDINAL;  (* Bitweises UND          *)
PROCEDURE or(x,y:CARDINAL):CARDINAL;   (* Bitweises ODER         *)
PROCEDURE not(x:CARDINAL):CARDINAL;    (* Bitweises NOT          *)
PROCEDURE set(VAR x:CARDINAL; y:CARDINAL);   (* Setzt Bit y in Zahl x   *)
PROCEDURE reset(VAR x:CARDINAL; y:CARDINAL); (* Loescht Bit y in Zahl x *)
PROCEDURE test(x,y:CARDINAL):BOOLEAN;  (* Testet Bit y in Zahl x  *)
PROCEDURE WriteBin(x:CARDINAL);        (* Schreibt Zahl binaer   *)
END Binary.

(* LISTING 5b *)
IMPLEMENTATION MODULE Binary;
FROM Terminal IMPORT Write;
PROCEDURE and(x,y:CARDINAL):CARDINAL;
BEGIN
RETURN CARDINAL(BITSET(x)*BITSET(y))
END and;
(* Schnittmenge, da nur gleich gesetzte Bits *)
(* übernommen werden *)
PROCEDURE or(x,y:CARDINAL):CARDINAL;
BEGIN
RETURN CARDINAL(BITSET(x)+BITSET(y))
END or;
(* Übernahme aller gesetzten Bits *)
PROCEDURE not(x:CARDINAL):CARDINAL;
BEGIN
RETURN CARDINAL(BITSET(x)/BITSET(65535))
END not;
(* Invertierung der Bits durch Subtraktion *)
(* des maximal möglichen Wertes *)
PROCEDURE set(VAR x:CARDINAL; y:CARDINAL);
VAR temp:BITSET;
BEGIN
temp:=BITSET(x);
IF y<16 THEN INCL(temp,y) END;
x:=CARDINAL(temp)
END set;
(* Setzen eines Bits durch Einfügen eines *)
(* Elementes in eine Menge *)
PROCEDURE reset(VAR x:CARDINAL; y:CARDINAL);
VAR temp:BITSET;
BEGIN
temp:=BITSET(x);
IF y<16 THEN EXCL(temp,y) END;
x:=CARDINAL(temp)
END reset;
(* Wie set, nur Entfernen des Elementes *)
PROCEDURE test(x,y:CARDINAL):BOOLEAN;
BEGIN
IF y<16 THEN IF y IN BITSET(x) THEN RETURN TRUE
                               ELSE RETURN FALSE
END
END test;
(* Erklärt sich selbst *)

PROCEDURE WriteBin(x:CARDINAL);
VAR y:BITSET;
   i:CARDINAL;
BEGIN
y:=BITSET(x);
FOR i:=15 TO 0 BY -1 DO   (* Ausgabe von der höchsten Stellen- *)
IF i IN y THEN Write('|') (* wertigkeit ab                     *)
         ELSE Write('0')
         END
END
END WriteBin;
END Binary.
```

Listing 5. Binary führt Operationen auf Bitebene aus

ST-Grafik durchleuchtet

Zur Grafikprogrammierung stellt das XBIOS (eXtended Basic Input Output System) im TOS nützliche Routinen zur Verfügung. Obwohl die Routinen sehr leistungsfähig sind, lassen sie sich leicht programmieren.

Die Line-A-Routinen benutzen den illegalen Opcode $AXXX der 68000-CPU. Die Routinen im XBIOS können von Assembler aus ausgerufen werden. Dazu werden von Assembler die Parameter und die Funktionsnummer auf den Stapel gebracht. Anschließend wird das XBIOS mit dem TRAP # 14-Befehl aufgerufen. Der Trap-Befehl stellt einen Softwareinterrupt dar. Danach sucht der Computer aus einer Tabelle die Adresse, an der die Routinen im XBIOS stehen. Nach der Rückkehr aus dieser Routine wird der Stapelzeiger korrigiert. Ergebnisse landen meist wieder im 68000-Register D0. Programmierer in C haben es hier einfacher. Die #include-Anweisung fügt die Datei »osbind.h« beim Compilieren hinzu, so daß die XBIOS-Routinen bequem als Funktionen aufgerufen werden können. Am Beginn des Programms muß also stehen:

```
#include <osbind.h>
```

Hier nun die grafikorientierten Funktionen des XBIOS:

XBIOS 2 - Physbase

Mit dieser Funktion wird die Startadresse der gerade dargestellten Bildschirmseite (physikalische Bildschirmbasis) ermittelt.

Assembler-Beispiel:

```
move.w   #2,-(sp)
trap     #14
addq.l   #2,sp
;D0 enthält die physikalische
Bildschirmbasis
```

C-Beispiel:

```
long start;
start=Physbase();
```

XBIOS 3 - Logbase

Logbase ermittelt die Startadresse der Bildschirmseite, auf der gezeichnet wird (logische Bildschirmbasis). Diese muß nicht der physikalischen Bildschirmbasis entsprechen. Es setzt also keinen großen Aufwand voraus, auf einer Seite zu malen, während der Monitor eine andere zeigt.

XBIOS 4 - Getrez

Mit Getrez läßt sich die Bildschirmauflösung feststellen.

2 = 640x400/ 2-Farben-Modus (monochrom)
1 = 640x200/ 4-Farben-Modus
0 = 320x200/16-Farben-Modus

Assembler-Beispiel:

```
        move.w      #4,-(sp)
;Funktionsnummer

        trap        #14

        addq.l      #2,sp
;Auflösung nun in D0
```

C-Beispiel:

```
        int  aufloesung
;aufloesung=Getrez();
```

XBIOS 5 - Setscreen

Um physikalische, logische Bildschirmbasis und/oder Auflösung zu setzen, eignet sich am besten diese Funktion. Verändert man allerdings die Bildschirmauflösung mit Setscreen in einer GEM-Applikation, so führt das zu Problemen, da die entsprechenden GEM-Parameter nicht von dieser Funktion gesetzt werden. Möchte man einen bestimmten Parameter nicht setzen, so gibt man ihm den Wert -1. Die Werte für die Auflösungen entsprechen denen von Getrez.

Assembler-Beispiel:

```
        move.w      #0,-(sp)
;Auflösung 320x200

        move.l      #$78000,-(sp)
;physische Basis

        move.l      #-1,-(sp)
;logische Basis unveränd.

        move.w      #5,-(sp)

        trap        #14

        add.l       #12,sp
```

C-Beispiel:

```
          long lbase,pbase;

          int  aufloesung;

          lbase=-1;
            /* unverändert */

          pbase=0x78000;

          aufloesung=0;
Setscreen(lbase,pbase,
aufloesung);
```

XBIOS 6 - Setpalette

Dieser Befehl »füllt« mit einem Funktionsaufruf alle Farbtöpfe neu. Als Parameter benötigt die Funktion einen Zeiger auf ein Feld von 16 Wörtern. Wort 0 entspricht dem neuen Inhalt von Farbtopf 0, Wort 1 für Topf 1 und so weiter. Nibble 2 der Wörter enthält den Rot-, Nibble 3 den Grün- und Nibble 4 den Blau-Anteil der Mischfarbe. »$0275« bedeutet also: Rot 2, Grün 7 und Blau 5 (ergibt ein Cyan).

Assembler-Beispiel:

```
        move.l      #pallete,-(sp)
;Startadresse

        move.w      #6,-(sp)
;Funktionsnummer

        trap        #14

        addq.l      #6,sp

        rts

pallete:        dc.w $000, $111,
$222, $333, $444, $555, $666
```

C-Beispiel:

```
setze_farben()

{
            int farben[15];

            ....
        /* Farbwerte ins Feld "
farben[]" */
        Setpallete(farben);
}
```

XBIOS 7 - Setcolor

Diese Funktion verändert den Inhalt von jeweils nur einem Farbtopf. Hierzu müssen als Parameter die Nummer des Topfes und der zu setzende Farbwert übergeben werden. Mit dieser Funktion ist es auch möglich, den Inhalt eines Topfes festzustellen. Man übergibt dann als Farbwert einfach den Wert -1.

Assembler-Beispiel: Weiß als Hintergrundfarbe

```
        move.w      #$777,-(sp)
;Neue Farbe

        move.w      #0,-(sp)
;Register 0

        move.w      #7,-(sp)
;Funktionsnummer

        trap        #14

        addq.l      #6,sp

        rts
```

C-Beispiel: Ermitteln der Hintergrundfarbe

```
int hintergrund()
{
    return(Setcolor(0,-1));
}
```

XBIOS 20 – Scrdmp

Hiermit läßt sich die über die ALT- und Help-Taste erreichbare Hardcopy-Routine vom Programm aufrufen. Sie erfordert keine Parameter.

Assembler-Beispiel:

```
    move.w    #20,-(sp)
    trap      #14
    addq.l    #4,sp
    rts
```

C-Beispiel:

```
hardcopy()
Scrdmp();
```

XBIOS 37 – Vsync

Bekannterweise baut sich das Monitorbild zeilenweise auf. Erfolgen nun Veränderungen des Grafikspeichers (Linien zeichnen etc.), so kann es zu unschönen Effekten, wie Flimmern, kommen. Die Vsync-Funktion schafft da Abhilfe. Sie wartet auf den nächsten Bildrücklauf, so lange also, bis der Bildschirmaufbau vollendet ist. Grafikausgaben lassen sich mittels dieser Funktion mit dem Strahlenrücklauf synchronisieren.

Assembler-Beispiel:

```
    move.w    #37,-(sp)
    trap      #14
    addq.l    #2,sp
    ...
```

C-Beispiel:

```
#define Vsync() xbios(37)
/* In fruehen osbind.h nicht
implementiert */
warte_raster()
{
Vsync();
}
```

Mit diesen Routinen lassen sich ohne große Anstrengungen wichtige Grafikparameter verändern. Von GEM-Applikationen aus sollte man allerdings nur auf die XBIOS-Routinen zurückgreifen, falls keine ähnlichen GEM-Routinen existieren. Die XBIOS-Routinen setzen nämlich keine GEM-Parameter. Ansonsten sind diese Routinen – richtig angewandt – sehr hilfreich in eigenen Programmen.

(F. Mathy/Udo Reetz/hb)

Spitzengrafik leichtgemacht

Die Farben und die Grafik des Atari ST können sich sehen lassen. Mit den Line-A-Routinen überzeugt diese Grafik auch noch durch komfortable Programmierung.

Einen Charaktermodus, wie der VIC im C64, besitzt der Videochip des ST, der Shifter, nicht, die Textdarstellung geschieht im Grafikmodus. Dies ist ohnehin erforderlich, da schon allein die Benutzeroberfläche den Grafikmodus erfordert. Deshalb ist es kein Wunder, daß das Betriebssystem eine Reihe leistungsfähiger Grafikroutinen beinhaltet. Viele C-Programmierer haben sicherlich schon die Grafikroutinen des GEM-VDI benutzt. Diese Art der Programmierung ist aber etwas umständlich und deshalb nicht besonders schnell. Die grundlegenden Grafikroutinen enthält der sogenannte »Line-A«-Teil des TOS . Das GEM-VDI bietet komplexere Routinen an, die dann die entsprechenden Line-A-Routinen aufrufen. Die direkte Anwendung der Line-A-Routinen bringt eine Geschwindigkeitssteigerung. Außerdem sind die Routinen flexibler anwendbar. Als ein Beispiel dienen die Spriteroutinen, die in der Spieleprogrammierung anwendbar sind, die das GEM-VDI aber nur zur Darstellung des Mauscursors verwendet. Die Line-A-Routinen lassen sich sowohl von Assembler als auch von C, Modula oder Pascal aus programmieren. GSTC-Besitzer finden eine geeignete Bibliothek im 68000er-Sonderheft, 6/86, ab Seite 114 (Werkzeugkasten für Superspiele) zusammen mit anderen nützlichen Funktionen. Gehen wir aber zunächst auf die Programmierung der Routinen in Assembler ein. Ihren Namen verdanken die »Line-A«-Grafikroutinen einer besonderen Eigenschaft der CPU MC 68000. Diese kennt zwei Gruppen von nicht implementierten Befehlen. Sie umfassen die Befehle mit den Opcodes, die mit den Nibbles $A und $F beginnen. Gehört der auszuführende Befehl dieser Gruppe an, so springt die 68000-CPU über die Vektoren $28 ($Axxx-Opcodes) beziehungsweise $2C ($Fxxx-Opcodes) in die gewünschte Exceptionroutine. Diese Exceptionroutine stellt nun anhand der restlichen drei Nibbles des Pseudo-Opcodes fest, wie der genaue Befehlscode aussah und führt die entsprechende Routine aus. Das TOS benutzt hier nur die $Axxx-Opcodes für die Grafikoperationen, die $Fxxx-Opcodes sind noch unbelegt und stehen somit dem Anwender zur Verfügung. Die Opcodes erreicht man also einfach durch Einsetzen des entsprechenden Worts in das Assemblerprogramm (DC.W $A000).

Im TOS sind 15 Line-A-Routinen implementiert, die die Pseudo-Opcodes von $A000 bis $A00E einnehmen. In diesem Beitrag befassen wir uns mit 12 dieser Routinen.

Beim Aufruf der Routinen ist darauf zu achten, daß unter Umständen die Inhalte einiger Daten- und Adreßregister verändert werden. Vor der Benutzung der Routinen muß eine Initialisierung stattfinden. Der Aufruf $A000 Init liefert in einigen Registern Zeiger auf wichtige Parameterblöcke. Eine Vielzahl von Parametern ermöglichen es dem Programmierer, die gewünschten Effekte zu erzielen. Die Register D0 und A0 enthalten die Startadresse des Line-A-Parameterblocks. Register A1 zeigt auf eine Tabelle, in der die Startadressen der Zeichensatz-Header stehen (der ST verfügt über drei Zeichensätze, den 6x6-, den 8x8- und den 8x16-Pixel-Zeichensatz). Der Line-A-Parameterblock umfaßt eine Vielzahl von Parametern. Die einzelnen Parameter erreichen Sie dann über einen bestimmten Offset. Das ist eine Zahl, die zur Startadresse des Parameterblocks addiert wird.

Hier eine Beschreibung der Parameter mit Angabe des Offsets und der Größe des Parameters (Wort/Langwort).

Zunächst zwei grundlegende, vom Grafikmodus abhängige Parameter:

0 Wort v__planes: Enthält die Anzahl der Bitplanes: Im 320x200-Modus sind es vier, bei 640x200 zwei und bei 640x400 ein Plane.

2 Wort v__lin__wr: Anzahl der Bytes pro Scanlinie. Bei 640x400 sind es 80, sonst 160 Byte. Es folgen fünf Zeiger auf wichtige Felder:

4 Lang contrl: Zeiger auf das contrl-Feld, das diverse Line-A Para-

meter enthält (wird bei den entsprechenden Routinen erklärt).

8 Lang intin: Zeiger auf das intin-Feld, in dem verschiedene Parameter vor Aufruf der Funktion abgelegt werden.

12 Lang ptsin: Zeiger auf das ptsin-Feld, in dem vor dem Aufruf der Funktion Koordinaten abgelegt werden.

16 Lang intout: Zeiger auf das intout-Feld, das nach Funktionsaufruf verschiedene Parameter enthält.

20 Lang ptsout: Zeiger auf das ptsout-Feld, das nach Funktionsaufruf verschiedene Koordinaten enthält. Die folgenden vier Parameter bestimmen die Zeichenfarbe:

24 Wort __fg__bp__1: Plane 0
26 Wort __fg__bp__2: Plane 1
28 Wort __fg__bp__3: Plane 2
30 Wort __fg__bp__4: Plane 3

Möchte man beispielsweise Farbe 7 haben, so geht man folgendermaßen vor: 7 dez. = 0111 binär. Deshalb:
__fg__bp__1=1 __fg__bp__2=1
__fg__bp__3=1 __fg__bp__4=0

Nun folgen verschiedene Parameter für die Zeichenoperationen:

32 Wort __lstlin: Muß beim Line-Befehl $FFFF sein.

34 Wort __ln__mask: Das Wort enthält das Linienmuster. Möchte man gerne eine gestrichelte Linie erhalten, wäre beispielsweise der Binärwert 0011001100110011 = $3333 zu empfehlen.

36 Wort __wrt__mode: Dies ist ein sehr wichtiger Parameter. Er bestimmt die Auswirkungen der Zeichenoperationen. Es gibt vier Modi:
0 = replace
1 = transparent
2 = exklusiv-oder
3 = invers-transparent

Der Replace-Modus ist der normale Zeichenmodus, hier wird die alte Grafik einfach »überpinselt«. Der Transparent-Modus mischt das bisherige Bild und die zuzufügende Grafik. Der Exklusiv-Oder-Modus setzt dort Punkte, wo sie bisher noch fehlten und löscht andererseits gesetzte Punkte. Im Invers-Transparent-Modus wird die hinzuzufügende Grafik invertiert und anschließend mit dem bisherigen Bild gemischt.

Die folgenden vier Parameter dienen bei den Routinen zum Zeichnen von Linien, horizontalen Linien und gefüllten Rechtecken.

38 Wort __x1: X1-Koordinate bei Line-,HLine- und Rechteck-Funktion
40 Wort __y1: Y1-Koordinate
42 Wort __x2: X2-Koordinate
44 Wort __y2: Y2-Koordinate

Die nächsten drei Parameter bestimmen das Füllmuster für die Funktionen zum Zeichnen von gefüllten Rechtecken und Polygonen:

46 Lang __patptr: Zeigt auf das Füllmuster, welches aus Worten besteht. Es ist also 16 Bit breit und beliebig hoch.

50 Wort __patmsk: Gibt an, wieviele Worte das Füllmuster hoch ist. __patmsk muß die Anzahl der Worte minus 1 enthalten.

52 Wort __multifill: Gibt an, ob das Füllmuster für eine (=0) oder für mehrere Planes (=1) ausgelegt ist. Die Line-A-Funktionen zum Zeichnen von gefüllten Rechtecken und Polygonen erlauben auch die Definition eines Bildschirmbereiches, in dem gezeichnet wird. Der Rest des Bildschirms ist für diese Operationen tabu. Leider arbeitet diese Möglichkeit, im Fachjargon »Clipping« genannt, nicht mit den anderen Line-A-Routinen, wie beispielsweise der Linien-Funktion, zusammen. Ist dies notwendig, so muß man wohl doch auf die GEM-VDI-Routinen zurückgreifen. Hier nun die Clipping-Parameter:

54 Wort __clip: Flagge, die anzeigt, ob Clipping an (=1) oder aus (=0) ist.

56 Wort __xmn__clip: X-Koordinate der linken oberen Ecke des Clipping-Rechtecks.

58 Wort __ymn__clip: Y-Koordinate dieses Punktes.

60 Wort __xmx__clip: X-Koordinate der rechten unteren Ecke des Clipping-Rechtecks.

62 Wort __ymx__clip: Y-Koordinate dieses Punktes.

Dies waren die wichtigsten Line-A-Parameter zur Grafikprogrammierung. Nun zu der Beschreibung der eigentlichen Line-A-Grafikfunktionen.

$A001 Put Pixel – Pixel setzen
Zu setzende Parameter:
(ptsin) x-Koordinate
(ptsin)+2 y-Koordinate
(intin) Farbe
sowie: __wrt__mode
Diese Funktion dient zum Setzen eines Bildschirmpunktes.

$A002 Get Pixel – Farbe eines Pixels bestimmen
Zu setzende Parameter:
(ptsin) x-Koordinate
(ptsin)+2 y-Koordinate
(intin) Farbe
Diese Funktion ermittelt die Farbe eines Bildschirmpunktes, welche man im MC68000-Register D0 zurückerhält.

$A003 Line – Zeichnen einer Linie
Zu setzende Parameter:
__x1,__y1,__x2,__y2 Koordinaten __fg__bp__1,__fg__bp__2,__fg__bp__3, __fg__bp__4 Farbe __ln__mask Linienmuster __wrt__mod Schreibmodus __lstlin muß auf $FFFF gesetzt sein. Diese Funktion dient zum Ziehen einer Linie von einem Pixel Breite.

$A004 Horizontal Line – Zeichnen einer horizontalen Linie
Zu setzende Parameter:
__x1,__y1,__x2 Koordinaten __fg__bp__1,__fg__bp__2,__fg__bp__3,__fg__bp__4 Farbe __wrt__mod Schreibmodus __patptr,__patmsk,__multifill Füllmuster

Mit dieser Funktion lassen sich sehr schnell horizontale Linien zeichnen. Außerdem ist es möglich, Füllmuster zu entwerfen. Mehrere $A004-Aufrufe bewirken so beispielsweise gefüllte Flächen. Die Rechteck- und die Polygon-Funktion, aber auch die Routine zum Zeichnen eines gefüllten Kreises im GEM-VDI, machen von dieser Routine Gebrauch.

$A005 Filled Rectangle – Gefülltes Rechteck zeichnen
Zu setzende Parameter:
__x1,__y1,__x2,__y2 Koordinaten
__fg__bp__1,__fg__bp__2,__fg__bp__3,__fg__bp__4 Farbe
__wrt__mod Schreibmodus
__patptr,__patmsk,__multifill Füllmuster
__clip,__xmn__clip,__ymn__clip,__xmx__clip,__ymx__clip Clipping
Der $A005-Opcode zeichnet ein gefülltes Rechteck. Auch hier ist ein Füllmuster möglich, ebenso wie Clipping. __x1/__y1 geben die Koordinaten der linken oberen Ecke an, __x2/__y2 die der rechten unteren.

$A006 Filled Polygon – Gefülltes Vieleck zeichnen
Zu setzende Parameter:
(ptsin) bis (ptsin)+2*n-1 X/Y-Koordinaten
(contrl)+2 Zahl X/Y-Paare
__y1 Zu füllendes y
__fg__bp__1,__fg__bp__2,__fg__bp__3,__fg__bp__4 Farbe
__wrt__mod Schreibmodus
__patptr,__patmsk,__multifill Füllmuster
__clip,__xmn__clip,__ymn__clip,__xmx__clip,__ymx__clip Clipping
Diese leistungsfähige Routine erlaubt das Zeichnen von gefüllten Vielecken (Polygonen). Die Koordinatenpaare werden (paarweise) im ptsin-Feld abgelegt. Jeder Aufruf verursacht jedoch jeweils das Zeichnen nur einer y-Zeile. Wollen wir zum Beispiel ein Dreieck mit den Koordinaten A (10/20), B (610/70), C (220/178), so ist die Routine 159mal für die y-Zeilen 20 bis 178 auszulösen. Wichtig ist, daß den Koordinatenpaaren ein weiteres folgt, das wieder auf die erste Ecke des Vielecks

zeigt. Bei obigem Dreieck müßten die Koordinaten deshalb folgendermaßen gesetzt sein:

```
(ptsin)      =   10 ;Punkt A
(ptsin)+2    =   20
(ptsin)+4    =  610 ;Punkt B
(ptsin)+6    =   70
(ptsin)+8    =  220 ;Punkt C
(ptsin)+10   =  178
(ptsin)+12   =   10 ;Punkt A
(ptsin)+14   =   20
(contrl)+2   =    3 ;3 Paare
```

$A007 Bitblock-Transfer – Sie zählt zu den grundlegenden Funktionen, die von der Textblock-Transfer- und Copy-Raster-Form-Routine benutzt werden.

$A008 Textblock-Transfer Diese Funktion dient der Textausgabe. Im Normalfall bietet sich ohnehin eine Textausgabe über das GEM-VDI an, da dies weitaus einfacher ist.

$A009 Show Mouse – Mauscursor aktivieren

Zu setzende Parameter:

(contrl)+2 = 0
(contrl)+6 = 1
(intin)

Diese Funktion dient zum Einschalten des Mauscursors.

$A00A Hide_Cursor – Wurde Hide_Cursor n-mal aufgerufen, so gilt dasselbe auch für Show_Mouse, um den Cursor zu reaktivieren. Setzt man (intin)=0, so wird der Cursor aber bereits beim nächsten Show_Mouse-Aufruf aktiviert.

$A00B Transform Mouse – Neue Mauscursorform setzen

Zu setzende Parameter:

(intin)	Aktionspunkt x-Koordinate, zum Beispiel Pfeilspitze
(intin)+2	Aktionspunkt y-Koordinate
(intin)+6	Maskenfarbe (normalerweise=0)
(intin)+8	Cursorfarbe
(intin)+10 bis (intin)+40	Hier wird die Form der Cursormaske definiert. Die Cursormaske ist 16 Pixels breit (1 Wort) und 16 Pixels hoch (daher 16 Worte).
(intin)+42 bis (intin)+72	Hier wird die Form des Cursors definiert. Das Format entspricht dem der Maske.

$A00C Undraw Sprite – Sprite löschen

Zu setzende Parameter:

Register A2: Zeigt auf den Sprite-Save-Buffer

Dies ist die erste der beiden Software-Sprite-Routinen im TOS. Der ST besitzt keine Möglichkeit zur hardwaremäßigen Erzeugung von Sprites. Die Undraw_Sprite-Routine dient zum Löschen eines gesetzten Sprites. Das 68000-Adreßregister zeigt hierbei auf den Sprite-Save-Buffer (SSB), der den durch das Sprite überdeckten Bildschirmbereich und einige Parameter enthält. Ein Sprite ist 16x16 Pixel groß. Die Größe des Sprite-Save-Buffers hängt von der eingestellten Bildschirmauflösung ab. Pro Plane sind 32 Worte zur Sicherung der überlappten Bildschirmdaten notwendig, hinzu kommen fünf Worte für andere Daten. Hieraus lassen sich für die drei möglichen Größen des SSB berechnen:

640x400- 2-Farben-Modus: 37 Worte
640x200- 4-Farben-Modus: 69 Worte
320x200-16-Farben-Modus: 133 Worte

$A00D Draw Sprite – Sprite darstellen

Zu setzende Parameter:

Register D0:	X-Koordinate
Register D1:	Y-Koordinate
Register A0:	Zeigt auf Sprite-Definition-Block (SDB)
Register A2:	Zeigt auf Sprite-Save-Buffer (SSB)

Diese Routine zeichnet ein Sprite an der Stelle X/Y. Hierzu müssen zudem A0 und A2 auf den Sprite-Definition-Block beziehungsweise den Sprite-Save-Buffer zeigen. Der SDB bestimmt das Aussehen des Sprites. Er ist 37 Worte lang und hat folgenden Aufbau:

Die ersten beiden Worte dienen dazu, einen Mittelpunkt des Sprites festzulegen. Haben wir zum Beispiel einen Pfeil mit der Spitze bei (4/6), so wird Wort 0 = 4 und Wort 1 = 6 gesetzt. Rufen wir nun die Draw_Sprite-Routine mit den Mittelpunktskoordinaten auf, so befindet sich anschließend die Spitze des Pfeiles genau im Mittelpunkt des Bildschirms.

Wort 0:	X-Offset. Hiermit läßt sich der Mittelpunkt des Sprites festlegen. Standardwert hierfür ist 8, da das Sprite 16 Pixels breit ist.
Wort 1:	Y-Offset. Standardwert hierfür ist 8. Das folgende Wort bestimmt den Setzmodus.
Wort 2:	0 = VDI-Format; 1 = XOR-Format
Wort 3:	Hintergrundfarbe
Wort 4:	Vordergrundfarbe

Nun folgen 32 Worte, welche die Form des Sprites in der gewohnten Bitmuster-Manier definieren. Allerdings folgen immer abwechselnd ein Wort des Hinter- und des Vordergrundmusters:

Wort 5:	Hintergrund-Muster Zeile 0
Wort 6:	Vordergrund-Muster Zeile 0
...	
Wort 35:	Hintergrund-Muster Zeile 15
Wort 36:	Vordergrund-Muster Zeile 15

Nun noch einige Worte zu den VDI- und XOR-Formaten: Im VDI-Format kann das Sprite maximal zwei Farben annehmen. Sind für das entsprechende Pixel sowohl im Hinter- als auch im Vordergrund-Muster die Bits gelöscht, so bleibt die Pixelfarbe erhalten. Ist nur das entsprechende Bit im Hintergrund-Muster gesetzt, so erscheint die Sprite-Hintergrundfarbe. Ist das entsprechende Bit im Vordergrund-Muster oder sind beide Bits gesetzt, so erscheint die Sprite-Vordergrund-Farbe. Hiervon unterscheidet sich das XOR-Format nur im vorletzten Fall: Ist nur das entsprechende Bit im Vordergrund gesetzt, so werden die Bits, die den Farbwert eines Bildschirmpunktes enthalten, mit den entsprechenden Bits der Sprite-Vordergrundfarbe (Wort 4) und/oder-verknüpft.

Doch nun genug der Theorie. Das Erlernte soll nun in einem Assemblerprogramm seine Anwendung finden. Als Beispiel nehmen wir das Zeichnen eines Rechtecks:

```
rect:      dc.w       $a000
;Init
move.w  #20,38(a0)
;_x1, linke obere Ecke
move.w  #39,40(a0)  ;_y1
move.w  #300,42(a0)
;_x2, rechte untere Ecke
move.w  #170,44(a0)  ;_y2
move.w  #1,24(a0) ;Farbe Bit 0
clr.w     26(a0)   ;Farbe Bit 1
clr.w     28(a0)   ;Farbe Bit 2
clr.w     30(a0)   ;Farbe Bit 3
clr.w     36(a0)   ;Replace-Modus
clr.w     54(a0)   ;kein Clipping
move.l  #must,46(a0)  ;_patptr
move.w  #1,50(a0)  ;2 Worte hoch
dc.w   $a005   ;Rechteck zeichnen
rts
;Ende
 must:     dc.w      $aaaa,$5555
;Füllmuster
```

Wie Sie sehen, lassen sich so auf einfache Weise selbst komplizierteste Figuren zeichnen. Auf jeden Fall lohnt es sich, einmal mit diesen interessanten Routinen zu experimentieren.

(F. Mathy/Udo Reetz/hb)

Ergänzen Sie jetzt Ihre HAPPY-COMPUTER-Sammlung

Schaffen Sie sich ein interessantes Nachschlagewerk und gleichzeitig ein wertvolles Archiv!

Kennen Sie alle »Happy Computer«-Ausgaben von 1985? Suchen Sie einen ganz bestimmten Testbericht? Oder haben Sie einen Teil eines interessanten Kurses versäumt? Suchen Sie nach einer speziellen Anwendung?
Damit Sie jetzt fehlende Hefte mit »Ihrem« Artikel nachbestellen können, finden Sie auf diesen Seiten eine Zusammenstellung aller wesentlichen Artikel der noch lieferbaren Ausgaben. Und so kommen Sie schnell an die gewünschten Ausgaben: Prüfen Sie, welche Ausgabe in Ihrer Sammlung noch fehlt, oder welches Thema Sie interessiert. Tragen Sie die Nummer dieser Ausgabe und das Erscheinungsjahr (z.B. 2/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein. Die ausgefüllte Zahlkarte einfach heraustrennen und Rechnungsbetrag beim nächsten Postamt einzahlen. Ihre Bestellung wird nach Zahlungseingang umgehend zur Auslieferung gebracht.

Stichwort	Titel	Seite/Ausgabe
	Aktuelles	
Computer	Amiga — ein Traumcomputer wird Wirklichkeit	9/10
	Atari: Lage gefestigt	14/11
	Der »Plus/4« ist endlich da	12/2
	Grundstein einer neuen Linie und kein zweiter PC	13/10
	Konsequentes Chaos (Der deutsche QL)	14/10
DFÜ	Akustikkoppler für C 64	9/1
	Ascom-Koppler jetzt auch für Atari	20/8
	Ein Anschluß unter dieser Nummer (Mailbox Nummern)	159/3
	Mailboxbetrieb in den USA	22/10
	Neues DFÜ-Programm für den Spectrum	22/10
	Nullmodem zum Aufstecken	12/1
Software	Atari-Schreiber jetzt für 520 ST	14/12
	Software fast zum Nulltarif	10/1
	Träume werden wahr (Schneider-Neuheiten aus England)	9/12
Drucker	Mac Inker, der sparsame Drucker	12/12
Floppy	Commodore-Floppy auf Trab gebracht	9/1
	Diskettenlaufwerk für den Sharp MZ-800	12/1
	Quick Disk — Die Floppy-Alternative (MSX)	20/4
Erweiterung	Mini-Expansion-Box für TI 99/4A	11/1
MSX	Das Musikwunder (Yamaha CX-5)	141/2
	Der Billig-MSX von Philips kommt	50/1
	CP/M mit MSX-Computer: so geht's	141/2
	Ein komplettes System von Philips	19/8
	Flotter Dreier (Sanyo, Goldstar und Canon)	23/5
	MSX-Mix	45/3
	Mit dem fliegenden Teppich auf Erfolgskurs	15/10
Bücher	Bücher zur DFÜ	111/3
	Bücher zum Denken (KI)	120/10
	Messeberichte	
	Die neuesten Heimcomputer (Winter-CES)	9/3
	Funkausstellung in Berlin: MSX war Trumpf	9/11
	Kampf der Kolosse (Winter-CES — Teil 1)	9/4
	Sommer-CES 1985: Weiche Welle in Chicago — Teil 1	9/8
	Software-Jackpot (Winter CES — Teil 2)	9/5
	Software-Super-Show in London (PCW-Show)	12/11
KI	Künstliche Intelligenz in Wiesbaden (AI Europa)	13/12
Musik	Musikmesse Frankfurt: Midi marschiert	22/5
	Interviews	
	David Crane (Ghostbusters Autor)	17/5
	Interview mit den »Print Shop«-Machern	14/8
	Jack Tramiel (Chairman Atari)	11/2
	Hardware-Tests	
Drucker	Bewußt robust (Europrint K 6311 FT)	31/5
	Software-Tests	
Textverarb.	Ein Textprogramm, das sich lohnt (Homeword/C 64)	77/4
	Drei Drucker im Test (STX 80, Gemini 10X, CP-80X)	16/1
	(Nachhall auf Seite 149 in 4/85) DWX 305: Schönschrift zum Niedrigpreis	18/2
	Eine heiße Verbindung (EP 22, EP 44, EXD 10)	26/5
	Kompakt und leise: Matrixdrucker GLP (Centronics)	24/1
	Regenbogenfarben — wie gedruckt (Okimate 20)	154/10
	Schön oder schnell (Horizon HX 80)	21/3
	Spectrum mit starken Typen (Gabriele 9009)	126/11
	Zwei Drucker für den Schneider (NLQ 401, GP 500 CPC)	112/8
Computer	Chinese mit britischem Paß (Triton 64)	22/2
	Der Musik Maestro (Yamaha CX-5)	28/4
	Der Neue: Commodore PC 128	46/5
	Der »neue« Spectrum	31/1
	Ein »Einsteiger« aus Taiwan (BIT-90)	16/2
	Joyce — Schneiders Einstieg in die Welt der PCs	24/11
	Quantensprung im Schneckentempo (QL dt. Version)	180/11
	Koreaner mit Deutsch-Talent (Ce-Tec/MSX)	18/3
	Schneiders neue Dimension (CPC 6128)	24/10
	Sharps Jüngster (Sharp MZ-800)	30/1
	Spectrum plus oder Spectrum minus	24/4
	Viel Computer für wenig Geld (Schneider CPC 664)	113/8
	YC-64: Fernöstlicher Biedermann (MSX Computer)	20/2
	Wer ist wer? (Atari 520 ST+ und 260 ST)	16/12
	Wie musikalisch ist mein Heimcomputer?	148/11
Laufwerke	3-Zoll-Erfahrungen (MCD-1-Floppy für Spectrum) (Discovery/Spectrum)	22/1
	Ein ungleiches Paar (Spectrum — VIC 1541 Interface)	21/4
	Lauf, Floppy, lauf! (SpeedDos plus/C64)	45/12
	Preiswertes Spectr um Floppysystem (Viscount System)	21/2
	Spectrum Diskettensystem im Plus-Look	20/3
	VC 1541 wird zur Rennfloppy	42/4
Recorder	Der Spectrum Sprinter (Datenrecorder: Sprint)	28/1
	Ein billiger Speicher für alle (Recorder MC 3810)	30/5
DFÜ	DFÜ auch mit dem TI (RS 232 für TI 99/4A)	25/5
	Kommunikation mit dem Spectrum	32/4
	Spartanisch aber gut (Ascom Akustikkoppler)	158/3
Sonstiges	Computer steuert Modelleisenbahn	176/11
	Der andere Weg (Spectrum Tastatur)	19/3
	Faszination der Technik (Fischer Technik Roboter)	44/11
	Famose Formel für den C64 (Formel 64)	40/12
	Grafpad Supergrafik für den Spectrum	16/3
	Haltet den Dieb (Alarmanlage für C 64, VC 20)	29/1
	Ohren oder Tasten? (Voice Command Modul/C 64)	40/10
	Peripherie für MSX (Plotter, 3½-Zoll-Floppy) (Joysticks im Vergleichstest)	26/1
	Roboter, Technologie der Zukunft (Fischertechnik)	45/4
	Starker Arm für Heimcomputer (Teach Robot)	38/4
	Tafelfreuden für Grafik-Gourmets (Atari Maltafel)	14/1/
	Vom Piepmatz zum Mini-Orchester (Spectrum Sound)	15/2
	Software-Tests	
Textverarb.	Ein Textprogramm, das sich lohnt (Homeword/C 64)	77/4
	Jedem seine Zeitung (The Newsroom)	118/8
	Schreiben ohne Frust	46/2
	Textverarbeitung für jedermann (Homewriter für MSX)	137/1
Sprachen	Basic-Erweiterung zum Spartarif (Aztec Basic/C 64)	76/4
	Drei Assembler für Atari-Computer im Vergleich	30/3
	Fortschritt rückwärts (CP/M-80 Emulator für 520 ST)	138/11
	Hisoft-Pascal jetzt Microdrive-kompatibel (Spectrum)	56/2
	Logo für den Atari 520 ST	134/11
	Mallard-80-Basic — ein starkes Stück	28/11
	Maschinensprache ist keine Zauberei (CPC 464)	107/8
	Prozessor-Welt von morgen: C 64 simuliert 68000	42/10
	Spezielles Spiele-Basic für den Spectrum	143/5
	Welches Basic für meinen MZ-700?	48/2
	Zwölf Farben in Mode 2 (Color Star für CPC 464)	110/8
Utilities	Das Programm, das Programme macht (Progressor)	33/5
	Disketten-Doktor für den C 128	42/12
	Quicksave für Spectrum	137/4
	SM-Kit — Das Werkzeug für Lehrling und Meister (C 64)	138/1
	Software-Knackern dazwischengepfuscht (Apple II)	27/3
Grafik	Beeindruckend (Print Shop — Druckprogramm)	50/2
	Die Maus bringt Farbe auf den Bildschirm (Apple)	52/2
	[illegible]	[illegible]

Stichwort	Titel	Seite/Ausgabe
	Koala Bilder zum Anfassen (Hardcopy-Programm)	57/2
	Mit dem Joystick programmiert (Designers Pencil)	140/5
	Viel Grafik für wenig Geld (Graphics Basic und Supergrafik 64 für C 64 im Vergleich)	44/2
	Vorsicht Kamera! (Take 1, Trickfilm Designer)	126/8
DFÜ	Apple II sucht Anschluß	154/3
	Contact 64 — Die Software zum Ascom-Koppler	142/5
	Spectrum auf Draht (DFÜ Vergleichstest)	124/8
Astronomie	Spectrums Sternstunden	34/3
	Sterngucker	158/10
Schach	Schachmatt per Telefon	156/10
	Spiele-Tests	
	Amazon	145/5
	Archon II: Adept	126/3
	Asylum	144/3
	Athletic Land	146/1
	A View to a Kill	169/10
	Ballblazer	167/10
	Boulder Dash	125/2
	Bounty Bob strikes back	139/8
	Cavelord	124/2
	Crazy Train	144/1
	D-Bug	118/2
	Deus ex Machina	146/4
	Don't buy this	168/12
	Doomdark's Revenche	148/5
	Dorodon	142/3
	Dragonsden	124/2
	Elektro Freddy	145/1
	Elite	164/10
	Eureka	144/4
	Fahrenheit 451	145/5
	Five-a-Side Football	166/10
	Formula One	140/8
	Frank Brunos Boxing	166/10
	Frankie goes to Hollywood	162/10
	Fruity Frank	145/4
	Ghettoblaster	169/11
	Ghostbusters	138/3
	Ghost Chaser	170/11
	Great American Cross Country Road Race	168/11
	Hacker	167/12
	H.E.R.O.	149/5
	Hyper Sports 1	143/3
	Karateka	146/4
	Kennedy Approach	168/12
	Knight Lore	143/3
	Macbeth	144/4
	Mask of the Sun	122/2
	Match Day	150/5
	Mindshadow	141/8
	Mr. Do	167/10
	Monster Trivia	168/10
	Nick Faldo plays the Open	169/11
	Nightshade	169/12
	Nodes of Yesod	169/12
	On Court Tennis	150/5
	Pitfall II	148/5
	Rama	145/5
	Rescue on Fractalus	168/10
	Rocket Ball	140/8
	Rockford's Riot (Bolder Dash II)	168/11
	Rock'n Bolt	139/8
	Sherlock Homes	121/2
	Seastalker	147/1
	Serpent's Star	142/4
	Software Star	165/11
	Spelunker	142/3
	Standing Stones	145/4
	Summer Games II	133/8
	Super Pipeline II	141/8
	The Ancient Art of War	149/5
	The Fourth Protocol	165/11
	The Hitchhiker's Guide to the Galaxy	138/4
	The Little Computer People Projekt	170/12
	The Way of exploding Fist	169/10
	Tour de France	170/11
	Where in the World is Carmen San Diego	163/11
	Whistler's Brother	141/3
	White Lightning	148/1
	Winter Games	164/12
	World Championship Boxing	170/12
	ZimSalaBim	141/3
	Spiele Tips	
	Abenteuer im Weltraum	152/5
	Amazon	172/10
	Aztec Challenge	147/4
	Aztec Tomb	147/4
	Aztec Tomb	173/10
	Beach Head	85/1
	Death in the Caribbean	142/8
	Death in the Caribbean	172/12
	Doomdark's Revenge	142/8
	Everyone's a Wally	173/10
	Ghostbusters	140/3
	Ghostbusters	147/4
	Ghostbusters	152/5
	Hampstead	172/12
	Heros of Karn	173/12
	Hexenküche	173/12
	Hobbit	146/3
	Hulk	143/8
	Hunch Back	85/1
	Karateka	172/12
	Lode Runner	174/11
	Lode Runner	174/12
	Mask of the Sun	173/11
	Masquerade	144/8
	Mindshadow	174/11
	Miner 2049er	147/4
	Pirate Adventure	126/2
	Pitfall	147/4
	Pitfall II	144/8
	Pitfall II	174/10
	Sabre Wulf	85/1
	Sands of Egypt	173/11
	Sands of Egypt	174/12
	Schloß des Grauens	152/5
	Secret Mission	173/11
	Ship of Doom	172/12
	Spelunker	144/8
	Strip Poker	152/5

Stichwort	Titel	Seite/Ausgabe
	Summer Games	
	Summer Games	
	Super Huey	
	The Dallas Quest	
	The Institute	
	The Institute	
	The Quest	
	The Witness	
	Time Maschine	
	Ultima II	
	Ultima II	
	Ultima III	
	Valhalla	
	Whistler's Brother	
	Zeppelin	
	ZimSalaBim	
	Zork	
	Listings	
Anwendung	Alle Neune (Jahresauswertung-Kegeln/C 64)	
	(Adreßverwaltung/C 64) Besseres Basic ganz einfach Software Basic 3.0/C 64)	
	Datenbank mit freiem Zugriff (C 64)	
	Der Halleysche Komet kommt (MSX)	
	Die Mini-Textverarbeitung (Spectrum)	
	Nachhall auf Seite 160 in 9/85	
	Do-it-yourself-Datenverwaltung (Mainfile II/C 64)	
	Einblick ins Innenleben (Disassembler/CPC 464)	
	Eine tolle Textverarbeitung für den Schneider (464)	
	Nachhall auf Seite 85 in 5/85	
	Funktionen optisch aufbereitet (VZ-200/Laser)	
	Geregelte Finanzen mit dem Commodore 64	
	Nachhall auf Seite 117 in 8/85	
	Morse-Decoder für Funkamateure (Spectrum)	
	Nebenkostenabrechnung (C 64)	
	Nachhall auf Seite 80 in 12/85 Optik mit Simons Basic (C 64)	
	Programme in Reih' und Glied (C 64)	
	Suchen, nein danke (Dateiverwaltung/CPC 464)	
	Nachhall auf Seite 117 in 8/85	
	Transistor-Schaltungen berechnen (L.d.M./Spectrum)	
	Turbo-Basic-Interpreter für Atari 800XL (L.d.M.)	
Grafik	Apple IIc-HiRes-Grafik auf dem Drucker	
	Bewegte Grafik mit drei Befehlen (CPC 464)	
	Farbspielereien (Atari)	
	Grafikentzerrung für Matrixdrucker (Spectrum)	
	Grafik-Window bekommt Nachwuchs (C 64)	
	Nachhall auf Seite 80 in 12/85	
	Grafikzauber (Apple II)	
	Hires Fantasy (C 64)	
	Rosetten-Grafik für den Spectrum	
	Schnelle Grafik aus dem Compiler (L.d.M./C 64)	
	Schöne schnelle Grafik (Grafik-Paket/C 64)	
	Solar-Painter (Spectrum)	
	Sprite-Editor (C 64)	
	Zauber der Farben mit Magic Painter (L.d.M./Atari)	
	Nachhall auf Seite 85 in 5/85	
	Zaubereien auf dem Bildschirm (L.d.M./Grafik/CPC 464)	
	Zeichenroutine für Kreise und Ellipse (CPC 464)	
	Zykloide für Grafiker und Mathematiker (C 64)	
	Nachhall auf Seite 79 in 12/85	
Spiel	Das Haus des Magiers (C 64)	
	Dasher, der Volltreffer (L.d.M./C 64)	
	Nachhall auf Seite 117 in 8/85	
	Der rasende Raider (C 64)	
	Diamantenfieber (L.d.M./Atari 48 KByte)	
	Nachhall auf Seite 85 in 5/85	
	Die Abenteuer eines rasenden Reporters (Report/C 64)	
	Geröllheimer (Atari)	
	Gespensterjagd im Schneider (CPC 464)	
	Nachhall auf Seite 85 in 5/85	
	Lumberjack Larrys Abenteuer in Bagdad (S.d.M/C 64)	
	Kalte Zeiten (Wintry Screen/C 64)	
	Kneipe zum hastigen Kellner (VC 20)	
	Mit dem Apple auf die Trainerbank (Aktion Apfelsaft)	
	Mit dem Atari-Computer auf Ölsuche (Atari)	
	Mücke mit Tücke (C 64)	
	Nachtflug (Spectrum)	
	Nachhall auf Seite 85 in 5/85 Niemandsland (C 64)	
	Pokerface für 16 KByte (Spectrum)	
	Psycho — die Macht des Geistes (C 64)	
	Nachhall auf Seite 80 in 12/85	
	Rennfahrer mit dem Joystick (Driver/C 64)	
	Rettet den letzten Baum (Insekt defense/C 64)	
	SAM — der Mann von der Baustelle (L.d.M./CPC 464)	
	Nachhall auf Seite 79 in 12/85	
	Schatzhöhle (Atari 800XL)	
	Vorsicht Hochwasser (Aquantor/L.d.M./C 64)	
	Über den Wolken (Flugplanung/C 64)	
	Wortsuchspiel (Spectrum)	
Tips&Tricks	AMPEL — grünes Licht für Atari-Maschinen-Programme	
	Auf Trap gebracht (CPC 464)	
	Basic bequem (C 64)	
	Basic-Compactor (Spectrum)	
	Nachhall auf Seite 80 in 12/85	
	Basic-Plus: Applesoft-Basic-Erweiterung (Apple II)	
	Beim C 64 piepst es	
	Bilder richtig konservieren (CPC 464)	
	Bildschirmtrick für den Commodore 64	
	Byte-Shifter (Spectrum)	
	Dateien hin- und hergerissen (Atari-IBM)	
	Der neue Checksummer ist da (C 64)	
	Der neue Checksummer ist da (C 64)	
	Der neue Checksummer (C 64)	
	Deutsche Sonderzeichen unter CP/M (CPC 464)	
	Die Maltafel wird zur Maus (Atari)	
	Nachhall auf Seite 49 in 11/85	
	Disk-Help für die schnelle Hilfe (Atari)	
	Disk- und DOS-Utility für alle Atari-Computer	
	Drei Tricks für MSX	
	Ein langes Gesicht für den C 64 (Longscreen 64)	
	Fehlerhilfe mit HELP & TRACE (VC 20)	
	Fensterkünstler (C 64)	
	Fettschrift für den 48 KByte-Spectrum	
	Find Label (Spectrum)	
	Flotte Primzahlen in Hisoft Pascal (Spectrum)	
	Fußball-Manager für Commodore 64	
	Grafik-Hardcopy in vierfacher Größe (C 64)	
	Kostenlose Speichererweiterung (C 64)	
	Listen leicht gemacht (C 64)	
	Make DATA für den Spectrum	
	Maschinencode-Routinen in Basic umgesetzt (CPC 464)	
	Microdrive-Aufwertung (Spectrum)	
	Nachhall auf Seite 79 in 12/85	
	Mondlandung (C 64)	
	Musik und Farbe (C 64)	
	Nie mehr Listingkummer mit dem Checksummer (C 64)	
	Nie mehr Listingkummer mit dem Checksummer (C 64)	
	Prima Werkzeug für den Programmierer (C 64)	
	Programmtransfer leicht gemacht (CPC 464)	
	Proportionalschrift für den Spectrum	
	Protokoll auf dem Drucker (CPC 464)	
	RAM-Disk für Atari 800XL	
	Ran an den Userport (C 64)	
	Renumber 64 (C 64)	
	Nachhall auf Seite 117 in 8/85	
	Rock me Amadeus (C 64)	
	RSX-Befehle ohne »@« (Schneider)	
	Schilderwald (Plakatschrift/C 64)	
	Schluß mit der Eintönigkeit (C 64)	
	Nachhall auf Seite 80 in 12/85	
	Spectrums COPY besser nutzen	
	Spectrumtasten mit Funktionen belegt (Spectrum)	
	Spectrum Tips & Tricks	
	Sprachkurs für Commodore-Basic (C 64)	
	Super-Merge für Commodore 64	
	Nachhall auf Seite 160 in 9/85	
	Super-Saver (C 64)	
	Statuszeile mit Uhr (Atari)	
	Tasword 464 mit DIN-Tastatur (CPC 464)	
	Texte auch im Grafikmodus (Atari)	
	Tippen mit dem Plotter (C 64)	
	Tips & Tricks rund um den Schneider	
	Töne aus dem Atari	
	Variablendump für Atari (Atari)	
	Variablen-Transfer (Spectrum)	
	Verflixter Listschutz (C 64)	
	Nachhall auf Seite 80 in 12/85	
	Vom Maschinencode zum Basic-Programm (C 64)	
	Wie die Bilder laufen lernten (Atari)	
	Zeilenakrobatik auf dem Schneider	
	Zwei SCREEN$ im schnellen Wechsel (Spectrum)	
	ZX81-Utility: Nützliches für Aufsteiger (C 64)	
	30 tolle Maschinencode-Routinen (Spectrum)	
	Grundlagen	
Speicher	Daten am laufenden Band	
	Daten auf der schnellen Scheibe	
	Floppy gegen Kassette	
	Selbst geschraubt ist halb gespart	
	So arbeitet das 1050-Laufwerk von Atari	
	So liest und schreibt die 1541	
	Speichermedium Endlosband	
	Tips, Tricks und Todsünden	
	Wohin in Zukunft mit Bits und Bytes	
Monitore	Farbmonitore — buntes Fenster zum Computer	
	Monitore: Richtig geplant, gekauft und genossen	
Drucker	Blitzsaubere Schrift mit Laserlicht (Laserdrucker)	
	Die »heißen« Drucker (Thermodrucker)	
	Farbspiele für Farbdrucker	
	Mit leisen Tönen (Tintenstrahldrucker)	
	Scharfe Nadel, spitze Typen (Matrix und Typenrad)	

Die Ausgaben 6/85, 7/85 und 9/85 sind bereits vergriffen und nicht mehr lieferbar!

Auch die bisher erschienenen Sonderhefte können Sie jetzt direkt bestellen:

SONDERHEFT 01/84: SINCLAIR
Unentbehrliche Informationen zu den Sinclair Computern ZX81 und Spectrum.

SONDERHEFT 01/85: SPECTRUM
Anwendungsbezogene Listings und Tips & Tricks für alle Spectrum-Fans.

SONDERHEFT 02/85: SCHNEIDER 1
Eine Fülle wertvoller Beiträge und Listings für alle Schneider-Anwender.

SONDERHEFT 03/85: SPIELE
Ein Super-Nachschlagewerk für alle Spiele-Fans mit 100 Spielen im Test und großer Marktübersicht.

SONDERHEFT 01/86: SCHNEIDER 2
Noch mehr Tips und Tricks für Einsteiger und Fortgeschrittene mit vielen interessanten Programm-Listings.

SONDERHEFT 02/86: ATARI 1
Besonders 800 XL- und 130 XE-Fans erwarten jede Menge Anwendungs- und Spiele-Listings sowie Informationen.

SONDERHEFT 03/86: 68000er
Umfassende Informationen zur neuen Computer-Generation und eine große Vergleichstabelle, die im Detail über alle 68000er informiert.

SONDERHEFT 04/86: SCHNEIDER 3
Eine Erweiterung für alle Schneider-Anwender, Super-Programm-Listings und großer Einsteiger-Teil.

SONDERHEFT 05/86: PROGRAMMIERSPRACHEN
Fuß fassen in »Pascal«, »C« und »Forth« mit jeweils einem grundlegendem Kurs und vielen Anwendungs-Listings.

SONDERHEFT 06/86: 68000er 2
Umfangreicher Listingteil, viele Informationen, Tips und Tricks für Anwender der 68000er-Computer.

SONDERHEFT 07/86: SCHNEIDER 4
Mit den Schwerpunkten Joyce und CP/M plus, Ratschlägen zur Vortex-Karte und vielen Tips & Tricks.

SONDERHEFT 08/86: COMPUTER ALS HOBBY
Wissenswertes für Einsteiger und zusätzliche Informationen zur Fernsehsendung Computerzeit.

Tragen Sie die Nummer des gewünschten Sonderheftes (z.B. 03/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein.

Am besten gleich mitbestellen: Die Happy-Computer-Sammelboxen



Für alle Leser, die »Happy Computer« regelmäßig kaufen, sammeln oder im Abonnement beziehen, gibt es ein interessantes Service-Angebot: die Happy-Computer-Sammelbox!

Mit dieser Sammelbox bringen Sie nicht nur Ordnung in Ihre wertvollen Hefte, sondern schaffen sich gleichzeitig ein interessantes und attraktives Nachschlagewerk. Ein kompletter Jahrgang (12 Ausgaben) paßt in eine der praktischen Sammelboxen!

Übrigens: Die Sammelbox ist nicht nur ein praktisches Aufbewahrungsmittel: Sie eignet sich auch hervorragend als Geschenk für Freunde und Bekannte zu vielen Anlässen.

Freie Kost für den Amiga

Jeder Computer-Besitzer träumt von Software, die nichts kostet und ohne Bedenken weitergegeben werden darf. Genau davon gibt es für den Amiga schon eine ganze Menge!

Nachdem der Amiga jetzt in den USA langsam Einzug in Schulen und Universitäten hält und hierzulande die »Freaks« auch schon kräftig Programme auf dem Amiga entwickeln, vergrößert sich das Angebot an Public-Domain-Programmen tagtäglich. Unter dem Begriff »Public-Domain« versteht man Programme, die von ihren Autoren zur kostenlosen Verteilung freigegeben wurden, damit jeder diese Programme nutzen kann. Es gibt also keinerlei Bedenken, wenn Sie Programme, die deutlich als Public-Domain gekennzeichnet sind, an Freunde und Bekannte weitergeben. Im Gegenteil, Sie handeln damit im Interesse des Autors, nämlich das Programm allen zugänglich zu machen.

Public-Domain-Programme sind eine gute Sache und im CP/M-Bereich längst bekannt und beliebt. Viele Programme erreichen schon den Standard von kommerzieller Software, womit auch das Hauptargument der Raubkopierer, sie würden kopieren, weil die Software zu teuer sei, nicht mehr gilt. Nun gibt es auch schon eine Menge freier Programme für andere Computer, aber noch nie ist eine Public-Domain-Welle so schnell ins Rollen gekommen wie beim Amiga.

Das zählt unbestreitbar zu den großen Verdiensten eines Mannes: Fred Fish. Er wohnt in Kalifornien und ist Initiator der »Amiga Software Library«, normalerweise nur unter der Bezeichnung »Fish Disks« bekannt. Er stellte bis jetzt 24 Disketten zusammen, die alle zu 80 bis 95 Prozent gefüllt sind. Wenn man sich überlegt, daß jede 3½-Zoll-Diskette auf dem Amiga 880 KByte faßt, ist das eine wahrhaft umwerfende Menge. Diese Programme schrieb er natürlich nicht alle selbst, obwohl er auch viele seiner eigenen Utilities auf den Disketten verewigte. Er stellt vielmehr in den USA eine Anlaufstelle für Programmierer dar, die Public-Domain-Software verbreiten wollen. Bei ihm stapeln sich alle Arten von Programmen, deren Autoren sich zu dem lobenswerten Schritt der Freigabe durchringen konnten.

Fred Fish stellte sie nach sorgfältiger Kompilierung und Prüfung inklusive Source-Code (die meisten Programme sind in C geschrieben und wurden mit Lattice- oder Aztec-C-Compiler compiliert) auf den Disketten zusammen, ordentlich in einzelne Subdirectories unterteilt. Welche Arbeit das macht, braucht wohl nicht extra erwähnt zu werden. Hier einige interessante Beispiele aus der großen Auswahl der »AmigaLibDisks«, wie Fred Fish sie selbst nennt.

Das interessanteste Programm für Grafik-interessierte Computer-Anwender enthält Diskette 21. Es nennt sich »Mandelbrot Set Explorer« (MSE) und ist mit Abstand das komfortabelste Mandelbrot-Programm für den Amiga. Es nutzt alle Auflösungsgrade des Amiga voll aus, die, wie auch die anderen Teile des Programmes, über Pull-down-Menüs angesprochen werden. Neben vielen weiteren Funktionen kann man auch zwischen zwei- und dreidimensionaler Darstellung wählen.

Für einen Punkt muß man Fred Fish besonderen Beifall zollen. Es gibt auf »seinen« Disketten kaum Programme, die keine Dokumentation enthalten. Jeder, der sich schon mit Public-Domain-Software beschäftigt hat, weiß, wie selten man zu solchen Programmen eine vernünftige Dokumentation bekommt.

Viele Programme, die auf den ersten Disketten veröffentlicht wurden und noch ziemlich primitiv waren, wurden inzwischen von anderen Leuten verbessert und sind auf neueren Fish-Disks zu finden. Die veröffentlichten Programme erfahren somit eine ständige Wartung und Verbesserung. Ein gutes Beispiel hierfür ist sicherlich das Programm »MicroEMACS«, ein sehr komfortabler Full-Screen-Editor, der auch auf vielen anderen Computern läuft. Dieser Editor wurde zunächst, zusammen mit anderen Programmen, auf Diskette 2 veröffentlicht und nimmt inzwischen eine ganze Diskette (Nummer 23) ein.

Auf Diskette 18 findet sich ein weiteres, sehr nützliches Programm. Es nennt sich »Scrimper«, eine Kurzform für »SCReen IMage PrintER«. Es wird als zusätzlicher Task, zum Beispiel vor einem Spiel, geladen und ist dann nur als schmaler Balken am unteren Rand des Bildschirms zu erkennen. Wählt man aus dem Menü von Scrimper »Print« aus, kopiert das Programm den aktuellen Bildschirminhalt in einen reservierten Speicherbereich und gibt eine exakte Hardcopy des Bildschirms auf einem Matrixdrucker aus. Man kann während des Druckens auf dem Bildschirm weiterarbeiten, ohne daß dies die Grafik irgendwie verändern würde; der Druck erfolgt nämlich aus dem Pufferspeicher. Das normale Preferences-Programm bestimmt die Größe, Lage und Helligkeit des Bildes. Sogar Farbausdrucke sind kein Hindernis.

Für Bastler bietet Diskette 18 eine interessante Umbauanleitung. In einer genauen Beschreibung erfährt man, wie der Amiga-Hauptprozessor MC 68000 durch einen MC 68010 ersetzt werden kann. Dadurch ergibt sich laut Anleitung ein Geschwindigkeitsvorteil von bis zu 50 Prozent! Der Amiga wird aufgeschraubt, der MC 68000 herausgenommen und der 68010-Prozessor hereingesteckt – und schon ist der Amiga schneller... wäre da nicht ein Problem. Der 68010 ist zwar vollkommen Pin-kompatibel zum 68000 und die meisten Befehle sind auch identisch. Lediglich bei einem Befehl, der beim 68000 ganz normal angesprochen wird, muß beim 68010 in den Supervisor-Mode gewechselt werden. Da dieser Modus normalerweise nicht aktiviert ist, kommt es zu einem Systemabsturz, sobald der 68010 auf diesen Befehl trifft. Dieses Problem löst aber die mitgelieferte Software (168 Byte kurz) mir nichts, dir nichts. Einfach das Programm laden und normal weiterarbeiten. Das kleine Programm fängt nun diesen Befehl (MOVE SR, <ea>) ab und wandelt ihn in einen äquivalenten Befehl für den 68010 um. Wirklich ein einfacher Umbau, den selbst ein unbedarfter Hobbybastler in wenigen Minuten erledigt.

Die Fish-Disks stellen beileibe nicht die einzige Public-Domain-Software für den Amiga dar. Auch sind die genannten Programme nur ein winziger Bruchteil des Angebots. Aber die Fish-Disks sind mit Abstand die am weitesten verbreiteten und wohl auch besten Public-Domain-Disketten, was die Dokumentation angeht. (Ottmar Röhrig/ts)

Info: Gegen einen geringen Unkostenbeitrag sind Fish-Disks unter anderem bei folgenden Firmen zu bestellen:
Deutschland: Interplan, Nymphenburger Str. 134, 8000 München 19
Schweiz: Softwareland AG, Franklinstr. 27, CH-8050 Zürich

OS9: Das Multitalent

Der Atari ST kann Multitasking! Man braucht nur das richtige Betriebssystem. Nun wurde das Multiuser-Multitasking-System OS9 auf den Atari St übertragen.

Welcher Besitzer eines Mikrocomputers hat sich nicht schon einmal ein zweites Gerät gewünscht, um nicht warten zu müssen, bis der Compiler endlich seine Arbeit beendet oder der Drucker ein zwölf Seiten langes Listing ausgedruckt hat? Das Drucken ist zwar nicht so kritisch, weil man sich mit einem Druckerspooler helfen kann. Doch was hilft ein Spooler, wenn man während einer Übersetzung schnell mal nachsehen möchte, ob man nicht doch eine Anweisung vergessen hat, oder wenn man sogar das Listing noch einmal ändern will?

Die Lösung für dieses Problem heißt Mehrprozeßbetrieb, besser bekannt unter der englischen Bezeichnung Multitasking. Leider war dieses Verfahren bisher nur den größeren und teureren Computern vorbehalten. Selbst multitaskingfähige Betriebssysteme für kleine Systeme, wie zum Beispiel Unix, benötigen zum Betrieb teure Hardware, wie schnelle Festplatten. »Hobbyisten« wollen oder können sich in der Regel so etwas gar nicht leisten.

Seit einiger Zeit gibt es jedoch den »Neuaufguß« eines längst vergessen geglaubten Betriebssystems für Mikrocomputer: OS9 von Microware. Ursprünglich für den Prozessor 6809 entwickelt, wurde es jetzt auf den 68000 übertragen. OS9 hält, was seine Leistungsfähigkeit angeht, im Vergleich zu den Betriebssystemen für andere Mehrprozeßrechner durchaus mit teureren Systemen mit. Es ähnelt äußerlich sehr Unix, ist aber etwas einfacher aufgebaut und stellt geringere Anforderungen an die Hardware. Eine Konfiguration aus einem Computer mit einem 68000-Prozessor, einigen Hundert KByte RAM, einem Terminal und einem oder besser zwei Disketten-Laufwerken reicht schon aus, um OS9 zu benutzen. Eine Festplatte macht das Leben allerdings auch unter OS9 sehr viel leichter.

Ein Mehrprozeß-Betriebssystem hat eine Reihe verschiedener Aufgaben.

Zum einen soll es dem Benutzer eine einheitliche, von der jeweiligen Hardware-Konfiguration unabhängige, Schnittstelle bereitstellen. Dafür gibt es bei OS9 Systemaufrufe, die das Programm mit der »TRAP #0«-Instruktion auslöst.

Eine Gruppe von Systemaufrufen steht für diverse Systemfunktionen, wie Speicher- oder Prozessorverwaltung, zur Verfügung. Die zweite Gruppe behandelt die Ein- und Ausgabe; die dritte Gruppe schließlich kann nur von Teilen des Systems, insbesondere von den Treibern, zur Ein- und Ausgabe benutzt werden.

Der Betriebssystem-Kern

Natürlich muß das Betriebssystem auch den Speicher verwalten. Einfache Betriebssysteme, wie CP/M, haben es da leicht: Ein CP/M-Programm auf dem 8080 wird einfach ab Adresse $100 von der Diskette in den Speicher geladen. Das ist kein Problem, denn es kann ja immer nur ein Programm laufen. Anders bei OS9. Hier laufen viele Programme gleichzeitig nebeneinander ab, und alle gerade auszuführenden Programme müssen im Speicher verfügbar sein. Dadurch liegen die Programme alle in unterschiedlichen Speicherbereichen. Bei größeren Computern erleichtert spezielle Hardware, die sogenannte Memory Management Unit (MMU), dem Programmierer die Handhabung mehrerer Programme. Das aktuelle OS9 braucht keine MMU, OS9 Level II, das eine MMU ausnutzt, ist allerdings angekündigt.

Das Problem besteht nun darin, daß ein Programm normalerweise so geschrieben und übersetzt wurde, daß es an einer ganz bestimmten Adresse, bei CP/M eben bei $100, beginnt. Absolute Referenzen auf Speicherbereiche und absolute Sprünge fallen daher weg. Glücklicherweise besitzt der 68000 einen sehr mächtigen Befehlssatz, der für die meisten Befehle auch eine sogenannte registerrelative Adressierung erlaubt. So kann zum Beispiel die Addition:

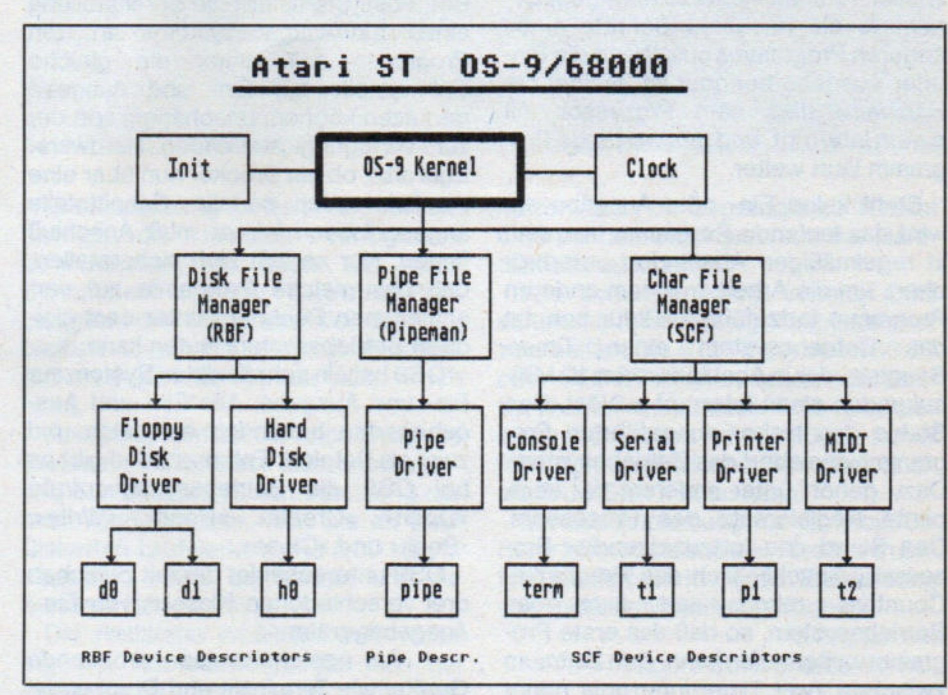
```
addq.l #1,$1000
```

auch unter Benutzung eines Adreßregisters durchgeführt werden:

```
addq.l #1,$1000(a6)
```

Damit wird die wirkliche Adresse, an der die Variable liegt, erst zur Laufzeit des Programms durch den Inhalt des Registers a6 bestimmt. Das Betriebssystem teilt jedem Programm zu Beginn Speicherplatz zu und informiert das Programm über die Anfangsadresse des Speichers in einem Adreßregister.

Das Programm selbst muß ebenfalls positionsunabhängig sein. Das erlaubt nur relative Sprünge. Diese Konvention betrifft die Hochsprachen-Program-



Aufbau des Multitasking-Multiuser-Betriebssystems OS9 beim Atari ST

mierer nicht, da die Compiler diese Arbeit übernehmen. Nur der Assembler-Programmierer muß darauf Rücksicht nehmen, aber fast jeder Assembler gibt bei absoluter Adressierung Warnungen aus.

OS9 vergibt den Speicher an sogenannte Module. Ein Spezialfall eines solchen Moduls ist der Codeteil eines ausführbaren Programms; andere Typen von Modulen sind Datenbereiche und Treiber für Ein- und Ausgabegeräte.

Jedes Programm besteht zunächst aus einem unveränderlichen Teil, dem Programmcode und einigen Konstanten. Von diesem Teil wird im Speicher nur ein einziges Exemplar benötigt, selbst wenn mehrere Benutzer das Programm starten. Den anderen Teil eines Programms, den Datenbereich, legt es bei jedem Aufruf neu an. Oder mit anderen Worten: Jeder Benutzer arbeitet in seinem eigenen Datenbereich.

Schließlich muß sich ein Mehrprozeß-Betriebssystem auch um die Verwaltung des Prozessors kümmern. Der Prozessor kann zu einem Zeitpunkt aber nur eine Anweisung ausführen, also werden die Programme – streng gesehen – doch nacheinander ausgeführt und zwar »verzahnt«, denn ein Programm beansprucht den Prozessor nicht die ganze Zeit. Zum Beispiel wartet er hin und wieder auf Eingaben von der Tastatur. Bei der eigentlichen Ein- oder Ausgabe bei Terminals und bei Disketten treten ebenfalls Wartezeiten auf, in denen der Prozessor keine Arbeit hat. Während dieser Zeit führt der Prozessor – anstatt nichts zu tun – gegebenenfalls die nächsten Schritte eines anderen Programms aus. Wenn die Ein- oder Ausgabe beendet ist, meldet die Hardware dies dem Prozessor mit einem Interrupt, und das wartende Programm läuft weiter.

Steht keine Ein- oder Ausgabe an, wird das laufende Programm trotzdem in regelmäßigen Abständen unterbrochen, um die Arbeit an einem anderen Programm fortzuführen. Dafür benutzt das Betriebssystem einen Timer-Baustein, der in Abständen von 10 Millisekunden einen Interrupt auslöst. Den Status des bisher ausgeführten Programms speichert das Betriebssystem. Dazu gehört unter anderem der komplette Registersatz des Prozessors. Den Stand des fortzusetzenden Prozesses, einschließlich des Programm-Counters, rekonstruiert dann das Betriebssystem, so daß das erste Programm weiterlaufen kann. Den Zeitraum zwischen zwei Timer-Interrupts nennt man auch eine Zeitscheibe.

Der Betriebssystem-Kern verwaltet eine Liste der Prozesse, die den Prozessor nicht benötigen, in einer sogenannten Warteschlange. In einer weiteren Schlange stehen die Prozesse, die auf Bearbeitung durch den Prozessor warten. Aus dieser zweiten Warteschlange wählt das Betriebssystem immer dann einen Prozeß zur Bearbeitung aus, wenn der bisher bearbeitete Prozeß den Prozessor nicht mehr benötigt, weil er beispielsweise auf eine Eingabe wartet oder eine Zeitscheibe abgelaufen ist.

Jedem Prozeß ist beim Start eine Basispriorität im Bereich von 0 bis 65535 zugewiesen. Für die Auswahl des nächsten auszuführenden Prozesses ist allerdings nur die aktuelle Priorität relevant. Der Prozeß mit der höchsten aktuellen Priorität kommt jeweils als nächstes an die Reihe. Am Ende einer Zeitscheibe reiht sich der bisher aktive Prozeß wieder mit seiner Basispriorität in die Prozessorwarteschlange ein, und die aktuelle Priorität aller übrigen wartenden Prozesse erhöht sich um Faktor 1. Dadurch kommen auch solche Prozesse zur Bearbeitung, die nur über eine sehr geringe Basispriorität verfügen.

Ein-/Ausgabesystem für einen universellen Einsatz

Als weitere wichtige Aufgabe eines Betriebssystems fällt die Bereitstellung eines Ein-/Ausgabesystems an. Ein Programm soll immer die gleiche Schnittstelle zur Ein- und Ausgabe benutzen können, unabhängig von der zur Verfügung stehenden Hardware: Egal also, ob ein Drucker nun über eine parallele oder serielle Schnittstelle angeschlossen ist, er muß Anschluß finden. Nur so läßt sich sicherstellen, daß das gleiche Programm auf verschiedenen Rechnern unter dem gleichen Betriebssystem laufen kann.

OS9 hat ein sehr flexibles System zur Ein- und Ausgabe. Alle Ein- und Ausgabegeräte behandelt es gleich und zwar als Dateien. Entsprechend gibt es bei OS9 die Betriebssystemaufrufe »Open«, »Create«, »Read«, »Write«, »Seek« und »Close«.

OS9 unterscheidet derzeit zwischen drei verschiedenen Klassen von Ein-/Ausgabegeräten:

– Auf zeichenweise arbeitende Geräte, wie Terminals und Drucker, ist nur ein sequentieller Zugriff möglich. Die Operation »Seek« ist hier also nicht erlaubt.

– Bei blockorientiert arbeitenden Geräten, also Disketten- oder Festplatten-Laufwerken, gibt es ein hierarchisches Dateisystem. Die Dateien haben Namen, die man beim Eröffnen einer Datei mit angibt. Ein Dateiname setzt sich aus dem Namen des Gerätes, dem Namen des Unterverzeichnisses, in dem sich die Datei befindet, und dem eigentlichen Dateinamen zusammen; zum Beispiel »/d0/USER/demo.program«.

– Pipes. Darauf wird später noch eingegangen.

Für jede dieser Klassen ist unter OS9 ein sogenannter Datei-Manager zuständig, der die Besonderheiten der jeweiligen Geräteklasse – zeichenweise Ein-/Ausgabe oder Dateisystem – berücksichtigt und an die einheitliche Schnittstelle anpaßt. Ein Datei-Manager ist ein Modul, das ständig im Speicher resident ist. Der Datei-Manager für zeichenweise arbeitende Geräte heißt »Sequential Character File Manager« (SCF), der für blockorientiert arbeitende Geräte »Random Block File Manager« (RBF).

Die Software zur Steuerung bestimmter Hardware befindet sich in Treibern (Device Driver), ebenfalls speziellen Modulen im Speicher. Die Treiber-Routinen ruft der jeweilige Datei-Manager auf. Sie sind für die eigentliche Ein-/Ausgabe verantwortlich. Bei einem Mehrprozeßsystem dürfen die Treiber natürlich keinesfalls in einer Schleife warten, bis Daten bereit sind. Sonst könnte in der Zwischenzeit kein anderer Prozeß arbeiten. Vielmehr benutzen die Treiber Interrupts und auch DMA (Direct Memory Access).

Treiber für spezielle Geräte kann ein geübter Programmierer auch selbst erzeugen; OS9 läßt im Prinzip beliebig viele Ein-/Ausgabegeräte mit gleichen oder verschiedenen Treibern zu. Die Identifizierung des gewünschten Gerätes beim Öffnen erfolgt durch einen Namen, der in einem Device Descriptor, wieder einem speziellen Modultyp, festgelegt ist. Der Device Descriptor enthält seinerseits Verweise auf den erforderlichen Treiber und den Datei-Manager.

Nach soviel Theorie folgt jetzt endlich das, was man bei OS9 als erstes wahrnimmt: die Benutzeroberfläche. Von anderen Betriebssystemen kennt man den Kommandointerpreter, der dort ein Teil des Betriebssystems ist. Anders bei OS9: Hier ist der Kommandointerpreter, die Shell, ein ganz normales Pro-

gramm. Man könnte sich also auch jederzeit seine eigene Shell schreiben.

Um nun überhaupt an so eine Shell zu kommen, muß man sich (je nach System-Konfiguration) erst mal »einloggen« – OS9 ist auch ein Mehrbenutzersystem! Wenn man seinen Benutzernamen und eventuell sein Paßwort angibt, landet man in einer Shell. Die Shell selbst kennt nur ganz wenige Kommandos, zum Beispiel eines, um seine Default Directory, also das Teilverzeichnis, in dem man arbeiten möchte, zu verändern. Alles, was man sonst eingibt, interpretiert es als Programmnamen. Die Shell versucht dann, ein Programm mit dem angegebenen Namen zu starten. Zunächst sucht es im Speicher, ob sich dort ein ausführbares Modul mit dem gewünschten Namen befindet. Wenn ja, wird es gestartet. Andernfalls sucht es eine Datei mit dem angegebenen Namen auf den peripheren Speichern. Statt der Shell-Kommandos gibt es an die 50 Hilfsprogramme. Viele dienen nur dazu, bestimmte Systemaufrufe zu benutzen, ohne dafür ein eigenes Programm schreiben zu müssen.

Ein wichtiger Punkt bei der Beschreibung eines Betriebssystems ist die Frage nach verfügbarer Software. Zu dem von Microware angebotenen System gehören, neben den vielen Utilities, ein Screen-Editor, ein Makro-Assembler und ein symbolischer Debugger, weiterhin C-, Pascal- und Fortran-77-Compiler und ein Basic-Interpreter sowie Netz-Software. Einige andere Firmen bieten ebenfalls Software, meist Compiler und Editoren, für OS9 an. Mit zunehmender Verbreitung dieses Betriebssystems wächst auch das Software-Angebot. Das unter GEMDOS bekannte ST-Pascal existiert inzwischen auch unter OS9. Es erzeugt sehr schnell ohne Temporärfiles, Assemblieren oder Linken ein ausführbares Modul und lädt den Editor im Falle eines Fehlers. Der Editor positioniert den Cursor auf die fehlerhafte Stelle und zeigt die Fehlermeldung in seiner Statuszeile an. Vom Editor aus ist dann der Compiler wieder zu starten. Damit kann man unter OS9 sehr schnell und bequem mit Pascal arbeiten.

Um OS9 auf dem Atari ST zu nutzen, benötigt man ein ROM-Cartridge mit dem sogenannten »Bootstrap Loader«. Hat man diesen in den Cartridge Port gesteckt und den Rechner eingeschaltet, so meldet sich anstelle von TOS jetzt der Bootstrap Loader. Nun lädt und startet man von der OS9-Systemdiskette oder von der Festplatte das System.

```
OS9/68000 Timesharing System   Level I  V1.2   86/04/01 18:51:23

User name?: user Password:
Process #03 logged on      86/04/01 18:51:31
Welcome!

shell version 1.2

$ procs -e  Id PId Grp.Usr   Prior  MemSiz Sig S  CPU Time   Age Module & I/O

  2    0     0.0         128  4.00k   0 w     0.24      ??? tsmon <>>> term
  3    5     1.0         128  4.00k   0 w     0.67     0:06 shell <>>> term
  4    6     1.0         128  8.50k   0 *     0.08     0:00 procs <>>> term

$ dir -er PASCAL                       Directory of PASCAL 18:59:55

 Owner  Last modified Attributes Sector Bytecount Name

 1.0   86/06/19 1854   ------wr          870      87 demo.pas
 1.0   86/05/10 2035   ------wr         205F     318 faks.pas
 1.0   86/05/25 2131   ------wr         2061    3503 pretty.pas

$ mdir  Module Directory at 19:28:51

kernel   systs    init     clock    scf
term     console  t1       sc68901  p1
printer  t2       sc6850   rbf      rb1772
d0       d1       ahdi     h0       ram
r0       pipe     pipeman  null     nil
shell    load     login    echo     dd
cio      dir      mdir     mfree    free
del      list     copy     procs    attr
pd       paslib   ed       tsmon

$ dir -e ! toupper >/p1 & +4
+5

$ procs -e  Id PId Grp.Usr    Prior      MemSiz Sig S     CPU Time   Age Module & I/O

  2    0     0.0     128  4.00k   0 w     0.22     ??? tsmon <>>>term
  3    2     1.0     128  4.00k   0 w     1.80    0:01 shell <>>>term
  4    3     1.0     128  4.75k   0 s     0.49    0:00 toupper <pipe >p1 >>term
  5    3     0.0     128  8.75k   0 s     0.36    0:00 dir <>>term >pipe
  6    3     1.0     128  8.50k   0 *     0.07    0:00 procs <>>>term

$ eof
```

Eine Beispielsitzung

Die Anpassung für den Atari berücksichtigt die meisten verfügbaren Schnittstellen, so auch die MIDI-Schnittstelle und den Festplatten-Port. Lediglich die Maus und der Joystick bleiben ungenutzt. Eine Grafikunterstützung gibt es derzeit auch noch nicht.

Ansonsten umfaßt die Anpassung die Tastatur – mit ladbarer Tastaturtabelle – und den Bildschirm, Unterstützung der nationalen Umlaute, etc., bis zu zwei Disketten-Laufwerke mit je 800 KByte Kapazität, sowie die serielle und die parallele Schnittstelle.

Die Festplatte erhält für OS9 einen eigenen Teilbereich, so daß man bei einer 20 MByte-Hard-Disk zum Beispiel je 10 MByte für TOS und für OS9 nutzen kann. Im Gegensatz zu TOS ist OS9 auf dem Atari auch von der Harddisk aus zu laden.

Wie schon erwähnt, vermag OS9 schon mit relativ wenig Hauptspeicher zu arbeiten. Die 512 KByte des 260 ST oder 520 ST reichen schon aus. Besser haben es natürlich die Besitzer eines Computers mit 1 MByte, zumal man den zusätzlichen Speicher auch als RAM-Disk nutzen kann.

Mit der Anpassung für die Atari-ST-Serie könnte erstmals ein breites Publikum am Fortschritt bei der Entwicklung von Betriebssystemen teilhaben.

(Jörg Lohse,
Ralph-Diether Marzusch/hb)

Info: Atari Corp (Deutschland) GmbH, Frankfurter Str. 89-91, 6096 Raunheim.



Animation in AmigaBasic

Mit mehr als 20 Befehlen für Animation trumpft das Basic des Amiga auf. Die Handhabung ist sogar für Neulinge auf diesem Gebiet ein leichtes. Lesen Sie hier, wie man's macht.

Die Fähigkeiten des Amiga, bewegte Bilder zu erzeugen, machen diesen Computer für viele Leute so attraktiv. Der Schlüssel dazu sind bewegte Objekte. Der Amiga kennt verschiedene Typen solcher Objekte:

1. Sprites: Durch Hardware (Spritelogik im Denise-Chip) über das normale Bild gelegte Objekte.
2. BOBs: Blitter Objects; per Software generierte spriteähnliche Objekte. Die Betriebssystemsoftware des Amiga nutzt zur schnellen Animation dieser Objekte die Fähigkeiten des Blitter-Chips.
3. AnimObjects: So nennt man Kombinationen von beiden. Die Animationsbibliotheken des Amiga erlauben es, Sprites und BOBs mit den gleichen Routinen zu behandeln. Das wiederum macht sich das Basic zunutze und gibt uns die Möglichkeit, Sprites und BOBs mit denselben Befehlen anzusprechen.

Die Basic-Befehle stellen nur einen kleinen Teil des Animationspotentials dar, das im Amiga steckt. Man kann damit jedoch eine ganze Menge anstellen, wie Sie sehen werden.

Der erste Schritt: Entwurf eines Objektes

In AmigaBasic legen wir ein Objekt – gleich ob Sprite oder BOB – in Form einer Stringvariablen fest. Der String sagt dem Basic-Interpreter, wie groß das Objekt ist, ob es ein Sprite oder BOB ist, wieviel und welche Farben es enthält, und natürlich wie es aussieht. Den genauen Aufbau dieser Strings brauchen wir jedoch nicht zu kennen. Der mitgelieferte Objekteditor nimmt uns nämlich diese Lernarbeit ab, wir entwerfen nur das Objekt. Das fertige Objekt speichert der Editor automatisch in der Form, die uns als String später von Diskette zur Verfügung steht. In der Regel greifen deshalb Basic-Programme mit Animationen nach Programmstart auf die Diskette zu. Lassen Sie also nach dem RUN-Befehl die entsprechende Diskette im Laufwerk.

Übrigens: Wer wissen will, wie eine Stringvariable, die ein AnimObject beschreibt, aussieht, der betrachtet am besten das Listing des Objekteditors genauer. Es besitzt eine vorbildliche Dokumentation.

Starten Sie nun den Objekteditor entweder aus der Workbench (OBJEDIT in der Basic-Demos-Schublade) oder vom Basic-Interpreter direkt mit LOAD "Basicdemos/Objedit" und RUN. Das Programm fragt nach dem Start zuerst, ob sie Sprites oder BOBs editieren wollen. Jeder der Objekttypen hat seine Vor- und Nachteile, die teilweise von der Hardware, teilweise aber auch vom Basic-Interpreter abhängen. BOBs sind langsamer als Sprites, dafür aber ist ihre Größe nur abhängig vom freien Speicher, wogegen die Sprites sich auf 16 Pixel Breite beschränken. BOBs können in beliebiger Anzahl auf dem Bildschirm erscheinen und das volle Farbenspektrum enthalten. Sprites sind auf drei (mit Hintergrund vier) Farben limitiert. Es lassen sich nur vier Sprites mit unterschiedlichen Farben zur selben Zeit auf derselben Bildschirmhöhe (auf denselben Rasterzeilen) zeigen.

AmigaBasic begrenzt für Sprites die Bildschirmtiefe des Hintergrundes auf zwei Bitplanes, also vier Farben. BOBs unterliegen dieser Beschränkung nicht. Für farbenfrohe Animation empfehlen sich BOBs; geht es mehr um Geschwindigkeit, greift man besser auf Sprites zurück. Für detailreiche Animationen wählt man eine sinnvolle Kombination aus Sprites und BOBs.

Geben Sie in unserem Beispiel eine Null für BOBs ein. Editieren Sie dann Ihr eigenes Objekt, wie Sie es sich vorstellen. Für unser Beispiel genügt ein kleines Kunstwerk (bezogen auf die Größe, versteht sich). Wie Sie sehen, ist die Farbwahl auf vier Farben beschränkt. Besitzen Sie einen 512 KByte-Amiga, können Sie Ihre Farbenfreude besser ausleben. Dazu ändern Sie im Objekteditor drei Zeilen, indem Sie die Hochkommas entfernen (Abkürzung für den REM-Befehl). Die Stelle ist im Listing dokumentiert.

Speichern Sie das Objekt unter dem Namen "Happyobjekt". Diesen Namen benötigen wir später wieder, wenn wir das Objekt im eigenen Programm verwenden. Der erste Schritt zur Animation ist eine Routine, die das Objekt von der Diskette holt. Das Objekt aktiviert dann der Befehl

```
OBJECT.SHAPE Nummer,Stringdefinition.
```

Die Stringdefinition des Objektes holen wir von der Diskette. Dies geschieht in der Form

```
OPEN "Happyobject" FOR INPUT AS 1
OBJECT.SHAPE 1,INPUT$(LOF(1),1)
CLOSE 1
```

Von diesem Zeitpunkt an steht das Objekt für unsere Animation bereit.

Arbeiten wir mit mehreren Objekten gleichen Aussehens, so wandeln wir den OBJECT.SHAPE-Befehl in seine zweite Syntax um:

```
OBJECT.SHAPE 2,1
```

Diese Befehlsvariante kopiert unser Blitter Object (Nummer 1) und weist ihm die Nummer zwei zu. Das zweite Objekt braucht dadurch keinen zusätzlichen Speicherplatz, denn beide Objekte greifen auf den gleichen Stringausdruck zu. Nun können wir schon zwei Objekte über den Bildschirm bewegen.

Bevor sich aber überhaupt etwas bewegt, legen wir zuerst die Startposition fest. Das geschieht durch folgende Befehle:

```
OBJECT.X Nummer,Wert
OBJECT.Y Nummer,Wert
```

In unserem Fall setzen wir Objekt 1 und Objekt 2 auf verschiedene Positionen des Bildschirms (siehe auch Listing 1). Die Positionsangabe kann einen Wert zwischen –32768 und 32767 annehmen. Arbeiten wir mit dem normalen Basic-Bildschirm (der dem Workbench-Screen entspricht), so muß der x-Wert zwischen 0 und 639 liegen, der y-Wert zwischen 0 und 199. Wählt man durch den SCREEN-Befehl eine andere Bildschirmauflösung (siehe Basic-Handbuch), darf der x-Wert bei Low-resolution höchstens 319, der y-Wert im Interlace-Modus bis zu 399 betragen.

Allein durch das Festlegen von unterschiedlichen Koordinaten könnten wir mit diesem Befehl bereits eine Animation erzeugen. Dabei müßten aber die Koordinatenänderungen durch das Basic-Programm berechnet werden. Wir wollen, während sich unser Objekt bewegt, unser Basic-Programm

weiterlaufen lassen. Die Animationsbefehle des AmigaBasic realisieren dies auch. Wir setzen lediglich die Geschwindigkeit und eine Bewegungsrichtung fest, der Rest geht (fast) von alleine. Doch noch einmal zu den Koordinaten der Objekte: Die Befehle »OBJECT.X« und »OBJECT.Y« sind in einer zweiten Form auch als Funktionen zu verwenden. Diese lauten wie folgt:
x = OBJECT.X (Objektnummer)
y = OBJECT.Y (Objektnummer)

Dies bietet immer eine Übersicht, wo sich unser Objekt im Augenblick befindet.

Nach der Positionierung unserer Sprites machen wir es sichtbar, wir schalten es an:
`OBJECT.ON Nummer (,Nummer....)`

In unserem Fall sollen beide Objekte, das Original und die Kopie erscheinen, also »OBJECT.ON 1,2«. Der Befehl »OBJECT.OFF« kehrt diese Anweisung um.

Um endlich zum Kern der Sache zu kommen, nämlich der Bewegung, benötigen wir die Angabe der Richtung und der Geschwindigkeit. Die Richtung geben wir ganz einfach dadurch an, daß wir eine Geschwindigkeit für die X-Richtung und eine Geschwindigkeit für die Y-Richtung festlegen:
`OBJECT.VX Objektnummer,Geschwindigkeit`

```
start:
 WINDOW 1,"Animation in AmigaBasic"
 WINDOW OUTPUT 1

eventtrapping:
 COLLISION ON
 ON COLLISION GOSUB abfrage

objektbestimmen:
 OPEN "basicdemos/ball" FOR INPUT AS 1
 OBJECT.SHAPE 1,INPUT$(LOF(1),1)
 CLOSE 1
 OBJECT.SHAPE 2,1

Bewegungfestlegen:
 OBJECT.X 1,10  'Startpositionen
 OBJECT.Y 1,10
 OBJECT.X 2,20
 OBJECT.Y 2,90
 OBJECT.VX 1,2  'Geschwindigkeit
 OBJECT.VY 1,1
 OBJECT.VX 2,3
 OBJECT.VY 2,2
 OBJECT.AX 1,2  'Beschleunigung Objekt 1 in x-
 OBJECT.AY 1,1  'und in y-Richtung
 OBJECT.AX 2,1  'und Objekt 2 genauso
 OBJECT.AY 2,2

Bewegung:
 OBJECT.ON
 OBJECT.START

WHILE 1  'endlosschleife
 IF OBJECT.X(1)>639 THEN OBJECT.X 1,1  'wenn durch zu grosse
 IF OBJECT.Y(1)>199 THEN OBJECT.X 1,1  'Geschwindigkeit nicht
 IF OBJECT.X(2)>639 THEN OBJECT.X 1,1  'kollidiert, dann zurueck
 IF OBJECT.Y(2)>199 THEN OBJECT.Y 2,1  'zum Anfang.
 IF OBJECT.VX(1)>180 THEN OBJECT.VX 1,1 'Durch x-Beschleunigung zu
                                        'schnell gewordenen Ball
                                        'langsam werden lassen.
 IF OBJECT.VY(1)>130 THEN OBJECT.VX 1,1 'ebenso in y-Richtung
 IF OBJECT.VX(2)>180 THEN OBJECT.VX 1,1 'und mit dem zweiten
Objekt
 IF OBJECT.VY(2)>130 THEN OBJECT.VY 2,1
WEND

abfrage:
  co = COLLISION(0)
  ob = COLLISION(co)
  vx = OBJECT.VX(co)
  vy = OBJECT.VY(co)

 IF ob>0 THEN
   vx=-vx
   vy=-vy
 ELSEIF ob=-4 THEN
   vx=-vx
 ELSEIF ob=-2 THEN
   vx=-vx

  ELSE
   vy=-vy
  END IF

 OBJECT.VX co,vx 'je nach Kollision
 OBJECT.VY co,vy 'Richtung aendern
 OBJECT.START

 RETURN
```

Listing 1. Ein Beispiel für Animationen mit BOBs in AmigaBasic

```
start:

WINDOW 1,"Scrolling is much fun"
WINDOW OUTPUT 1

Bildfuell:

DIM area.pat%(3)
area.pat%(0)=&H1111       'Flaeche mit
area.pat%(1)=&H2222       'Muster
area.pat%(2)=&H3333       'fuellen
area.pat%(3)=&H4444
PATTERN &HFFFF,area.pat%
AREA (1,1)
AREA (600,1)
AREA (600,180)
AREA (1,180)
AREAFILL

scrolling:

SCROLL (1,1)-(300,90),2,3          'vier
SCROLL (300,1)-(600,90),-2,3       'Scroll-
SCROLL (1,91)-(300,180),2,-3       'Bereiche
SCROLL (300,91)-(600,180),-2,-3    'festlegen

  x=x+1
  IF x<31 THEN
   GOTO scrolling
  ELSE
   LOCATE 12,35
   PRINT "peng!"
  END IF
```

Listing 2. Mit wenigen Befehlen erzielt man tolle Scrolling-Effekte

`OBJECT.VY Objektnummer,Geschwindigkeit`

Dabei wird die Geschwindigkeit in Anzahl der Pixel pro Sekunde gemessen. Wie schnell die richtige Geschwindigkeit für eine stufenlose Animation ist, unterscheidet sich von Fall zu Fall. Beim Programmieren muß man damit so lange herumexperimentieren, bis das Ergebnis zufriedenstellt.

Eine stetige Animation ist aber noch nicht alles. Um auch Beschleunigungs-Effekte in eigene Programme einbauen zu können, sind im AmigaBasic zwei weitere Befehle enthalten. Diese Befehle vermeiden, daß unser nebenher laufendes Basic-Programm sich mit der Geschwindigkeitsänderung herumschlagen muß. Der Computer erhöht oder vermindert das Tempo der Objekte mit den Befehlen
`OBJECT.AX Objektnummer,Wert` und
`OBJECT.AY Objektnummer,Wert`
wobei »Wert« die Beschleunigungsrate in »Pixel pro Sekunde im Quadrat« angibt. Ist die Beschleunigung Null oder wurde keine Beschleunigungsrate festgelegt, bleibt die anfangs gewählte Geschwindigkeit unverändert erhalten. Diese Beschleunigungsbefehle ermöglichen auch, physikalische Vorgänge zu simulieren, ohne jedesmal die Geschwindigkeit neu berechnen zu müssen.

Oft ist es notwendig, die Geschwindigkeit eines bewegten Objektes zu überwachen. Dazu dient die zweite Syntax der Geschwindigkeitsbefehle als Funktion:
vx = OBJECT.VX (Objektnummer)
vy = OBJECT.VY (Objektnummer)

Ist die so festgestellte Geschwindigkeit höher als der gewünschte Wert, können Sie entsprechend reagieren und das Objekt beispielsweise abbremsen.

All das, was wir bisher besprochen haben, funktioniert jedoch nur, wenn wir die Animation starten. Mit den Positions-, Geschwindigkeits- und Beschleunigungsbefehlen setzen wir nämlich nur fest, wie sich alles theoretisch bewegt. Erst nach dem Anschalten des Objekts und somit der Animation wird es auf unserem Bildschirm lebendig. Die Bewegung leitet der Befehl
`OBJECT.START Nummer (,Nummer,...)`
ein. Ohne Angabe einer Nummer starten alle Objekte, die im aktiven Output-Window liegen. Die Bewegung stoppen wir mit dem Gegenbefehl
`OBJECT.STOP Nummer (,Nummer,..)`

Wo sich viel bewegt, kommt es aber auch leicht zu Kollisionen. Immer, wenn ein Zusammenstoß passiert, stoppt Amiga-

Basic die Bewegung der beiden kollidierenden Objekte. Dasselbe geschieht, wenn ein Objekt die Window-Grenzen berührt. Deswegen ist es oft notwendig, die Bewegung mit »OBJECT.START« wieder in Gang zu setzen. In den meisten Fällen beendet aber eine kleine, spektakuläre Explosion die Szene...

Doch vor allen diesen Maßnahmen muß das Basic eine Kollision erst einmal erkennen. Dazu dient das sogenannte »Event-Trapping«. Als Event, also als Ereignis, gilt in unserem Falle eine Kollision. Das Event-Trapping aktiviert der Befehl COLLISION ON.

Ausgeschaltet wird es durch COLLISION OFF.

COLLISION STOP nimmt zwar die Kollisionen wahr und reiht sie in seine »Event queue« ein (eine Liste, in der die Kollisionen gespeichert sind; bis zu 16 Kollisionen kann sich der Basic-Interpreter merken), führt aber keine Sprünge in Unterroutinen aus, die wir durch
ON COLLISION GOSUB label
aktivieren. Ist also das Kollisions-Trapping angeschaltet, springt unser Programm automatisch an die richtige Stelle.

Mitunter kommt es vor, daß sich einige Objekte berühren, ohne daß das von Bedeutung ist, also festgestellt werden soll. Mit dem Amiga läßt sich festlegen, welches Objekt mit welchem Objekt kollidieren darf. Die Anweisung mit der Syntax

```
OBJECT.HIT Objektnummer,MeMask,Hitmask
```

setzt diese Spezifikationen durch sogenannte Masken fest.

Diese »Masken« sind Werte, deren binärer Wert dem Computer sagt, welche Objekte sich berühren dürfen. Die Masken »MeMask« und »HitMask« sind beide jeweils 16 Bit breit. MeMask beschreibt das Objekt, das durch den »OBJECT. HIT«-Befehl angewählt wurde, HitMask dasjenige, mit dem es zusammenstößt (die Namen der Masken entstammen dem ROM-Kernel-Manual und wurden auch ins Basic übernommen). Der Amiga führt bei einer Berührung ein logisches UND zwischen der MeMask des einen Objektes und der HitMask des anderen Objektes durch. Dadurch erkennt der Computer, ob die Kollision von Bedeutung ist oder ob sie übergangen wird. Ist das niederwertigste Bit in HitMask gesetzt, wird eine Kollision mit den Window-Grenzen registriert. Die restlichen Bits dienen zur Feststellung von Kollisionen zwischen den verschiedenen Objekten.

Das AmigaBasic-Handbuch gibt hier allerdings nicht sonderlich viel Auskunft. Es verweist lediglich auf das ROM-Kernel-Manual. Daher beschreiben wir hier die Funktion der Masken etwas näher. Nehmen wir als Beispiel drei Objekte aus einem Asteroids-Spiel: Objekt 1 ist ein Asteroid (durch Kopieren mit dem »OBJECT.SHAPE«-Befehl erhalten wir mehrere davon), Objekt 2 ein Raumschiff und Objekt 3 ein Schuß des Raumschiffs.

Der Befehl »OBJECT.HIT 1,8,7« bewirkt, daß Kollisionen eines Asteroiden mit der Window-Grenze, dem Schiff und den Schüssen festgestellt werden können. »OBJECT.HIT 2,2,9« sagt dem Computer, daß ein Schiff mit der Window-Grenze und einem Asteroiden kollidieren kann. »OBJECT.HIT 3,4,9« bewirkt, daß Schüsse mit der Window-Grenze und den Asteroiden kollidieren können, daß ein Zusammenstoß zwischen Schüssen und dem Raumschiff jedoch nicht zu registrieren ist. (Haben Sie sich schon mal selbst abgeschossen?)

In diesem Beispiel entsprechen die MeMasks der Objekte 1, 2 und 3 den Werten 8, 2 und 4. In Binärwerten ausgedrückt: Objekt 1 entspricht Bit 3, also 2^3 = 8, Objekt 2 entspricht Bit 1 und Objekt 3 Bit 2. Die HitMask jedes Objektes beschreibt die Objekte, mit denen das entsprechende Objekt kollidieren kann. Bit 0 ist hier jedesmal gesetzt, da jedes der Objekte mit der Windowgrenze kollidieren soll. Die restlichen Bits entsprechen wie in der MeMask dem jeweiligen Objekt.

Durch diese Technik legt man nun genau fest, welches Objekt mit welchem zusammenstoßen darf. Beispiel: Die Asteroiden sollen einfach aneinander vorbeifliegen. Wenn wir die MeMask des Asteroiden mit seiner HitMask durch ein logisches UND verknüpfen, resultiert daraus eine Null. Das bedeutet, Asteroiden können nicht miteinander kollidieren.

Räumliche Grafik beeindruckt

Wie schon gesagt, springt ON COLLISION sofort in eine Unterroutine, wenn eine Kollision stattfindet (und wenn das Event-Trapping mit COLLISION ON aktiviert ist). In dieser Unterroutine wird all das behandelt, was bei einer Kollision geschieht. Also entweder eine Explosion, ein Bewegungsstopp, oder ähnliches. Dazu müssen wir allerdings wissen, welches Objekt einen Crash gebaut hat. Das findet die Funktion COLLISION (0) heraus. COLLISION (0) gibt die Nummer des zuletzt kollidierten Objektes bekannt. Danach fragen wir mit »COLLISION (Objektnummer)«, womit dieses Objekt zusammengestoßen ist. Diese Funktion teilt die Nummer des zweiten in die Kollision verwickelten Objektes mit, oder aber die entsprechende Seite bei Kollisionen mit den Window-Grenzen: -1 für oben, -2 für links, -3 für unten und -4 für rechts. »COLLISION (-1)« ergibt die Nummer des Windows, in dem die Kollision stattfand – falls wir gleichzeitig mehrere Windows aktiviert haben (wie im Beispiel "Basic-demos/Demo", das sich auf der Basic-Diskette befindet).

Der Basic-Interpreter merkt sich bis zu 16 Kollisionen in einer sogenannten »queue« (wörtlich übersetzt »Schlange«), also einer 16 Zahlen langen Reihe. Wenn 16 Kollisionen stattgefunden haben, wird keine weitere in die Liste aufgenommen. »COLLISION (Objektnummer)« nimmt jeweils die oberste Zahl vom Stapel und schafft so wieder Platz für die nächste Kollision. »COLLISION (0)« beläßt sinnvollerweise die Zahlen auf dem Stapel, denn wir müssen für dieselbe Kollision die COLLISION-Funktion ja nochmal aufrufen, um das andere Objekt herauszufinden.

Das Stapel-Prinzip ist in einer langsamen Interpretersprache wie Basic notwendig, da unser Programm bei schnell aufeinanderfolgenden Kollisionen nicht mitkommt. Damit wir keine der Kollisionen verpassen, hat man dem Programmierer diese Hilfe gegeben.

Bewegung macht jedes Programm schöner. Sie allein reicht aber für wirklich beeindruckende Animationen noch nicht aus. Räumliche Grafik muß dazukommen. Erst sich überdeckende Objekte erzeugen einen dreidimensionalen Effekt. Auch da läßt uns das Basic nicht im Stich. Der Befehl

```
OBJECT.PRIORITY Objektnummer,Priorität
```

weist jedem Objekt eine Priorität zu. Der Wert kann jeden Wert zwischen -32768 und 32767 annehmen. Passieren zwei Objekte die gleiche Stelle des Bildschirms, überlagert das Objekt mit der höheren Priorität dasjenige mit der niedrigeren.

Diese Prioritätsbestimmung gilt jedoch nur für BOBs, denn Sprites haben eine automatische Priorität: Sprite 1 überlagert Sprite 2, Sprite 2 überlagert Sprite 3, und so weiter.

Manchmal stört es, wenn ein BOB oder Sprite wichtige Stellen verdeckt. Das sogenannte Betriebssystem-Clipping hilft, dies zu vermeiden. Der Syntax für diese Prozedur lautet:

```
OBJECT.CLIP (x1,y1)-(x2,y2)
```

Alle Objekte, deren Position außerhalb des so fixierten Bereichs liegt, verschwinden automatisch.

Wenn wir unsere Objekte nicht mehr brauchen, wollen wir den Speicher für das restliche Programm – oder für neue Animationen – freigeben. Das erreichen wir durch

```
OBJECT.CLOSE Objektnummer
```

Mit Basic läßt sich also schon eine ganze Menge erreichen! Für diejenigen, die noch tiefer einsteigen wollen, gibt es das LIBRARY-Statement, mit dessen Hilfe man auf die ROM-Routinen des Amiga zugreifen kann (Intuition-Funktionen, Workbench-Routinen und Grafik-Bibliotheken). Zu diesem Thema lesen Sie in einer der nächsten Ausgaben des 68000er-Magazins mehr.

AnimObjects, so die Bezeichnung für Kombinationen aus Sprites und BOBs, unter derselben Software-Ansteuerung, ermöglichen besonders effektvolle Grafiken. Oft spielt aber auch eine gute Hintergrundgrafik eine große Rolle, wenn es darum geht, Eindruck zu machen. Wenn sich dann da auch noch etwas rührt, haben wir das Nonplusultra.

Die Befehle GET und PUT erlauben das Kopieren von beliebigen Bereichen der Hintergrundgrafik. Mit

```
GET (x1,y1)—(x2,y2),Arrayname
```

legen wir einen rechteckigen Grafikbereich in einem Variablenfeld ab. Mit Hilfe des PUT-Befehls setzen wir diesen Bereich an eine andere Stelle. Der Befehl dazu lautet:

```
PUT (x,y),Arrayname,Aktionsverb.
```

Das Aktionsverb beschreibt die Art, in der die Grafik in den Speicher gesetzt wird: PSET setzt alle Punkte, PRESET löscht sie, und die Worte AND, OR und XOR verknüpfen die im Array abgelegte Grafik logisch mit der Hintergrundgrafik dieser Stelle. Wenn wir das Aktionsverb nicht angeben, ist der Defaultwert XOR.

Ein Beispiel für die Funktionsweise der GET- und PUT-Befehle finden Sie auf Ihrer Basic-Diskette unter "Basicdemos/Picture".

Besonders schöne Effekte ruft der Scroll-Befehl hervor. Dabei handelt es sich allerdings nicht um das vielgerühmte Playfield-Scrolling des Amiga, sondern um ganz normales HiRes-Scrolling (der Großteil der AmigaBasic-Befehle ist identisch mit Macintosh-Microsoft-Basic 2.1, das ja auf den Amiga umgeschrieben wurde). Dabei bewegt sich nur die Grafik innerhalb eines bestimmten Bereiches, ohne die daneben liegende Grafik mitzuscrollen. Die Syntax lautet:

```
SCROLL (x1,y1)—(x2,y2),delta—x,delta-y
```

Die Delta-Angaben sagen dem Computer, um wieviele Pixel nach rechts und um wieviele nach unten gescrollt werden soll. Die Angabe negativer Zahlen ergibt beim x-Wert eine Bewegung nach links, beim y-Wert eine Aufwärtsbewegung.

Einen schönen Effekt, den Sie sicherlich vom AmigaTutor her kennen, zeigt unser Beispiel-Listing für den Scroll-Befehl. Der gesamte Bildschirminhalt wandert in die Mitte des Bildschirms. Eine andere hübsche Anwendung des Scroll-Befehls läßt sich durch die Vibration eines bestimmten Teiles der Grafik durch abwechselndes Auf- und Abscrollen erzeugen: zum Beispiel ein frierender Eisbär oder ein Erdbeben.

Auch in den normalen Grafikbefehlen wie LINE, PSET, CIRCLE oder PAINT steckt ein gewisses Animationspotential. Besonders schnell ist die Grafik gefüllter Polygone mit den Befehlen AREA und AREAFILL. Experimentieren Sie nach Belieben. Vielleicht erzielen Sie einen sensationellen Effekt. Dann schreiben Sie uns doch. Die Redaktion »Happy-Computer« freut sich über alle Einsendungen!

(Manfred Kohlen/ts)

Die wichtigsten Animationsbefehle des AmigaBasic

Befehle

OBJECT.SHAPE Objektnummer,Stringausdruck
Festlegen des Objekts. Der Stringausdruck bezeichnet Größe, Art (Sprite oder BOB), Farbanzahl, etc. Er wird durch den Objekteditor festgelegt.

OBJECT.SHAPE Objektnummer2,Objektnummer1
Kopiert Objekt 1 nach Objekt 2

OBJECT.X Objektnummer,Position
Legt aktuelle X-Koordinate des Objekts fest.

OBJECT.Y Objektnummer,Position
Legt aktuelle Y-Koordinate des Objekts fest.

OBJECT.VX Objektnummer,Wert
Bestimmt die Geschwindigkeit des Objekts in X-Richtung.

OBJECT.VY Objektnummer,Wert
Bestimmt die Geschwindigkeit des Objekts in Y-Richtung.

OBJECT.AX Objektnummer,Beschleunigung
Geschwindigkeitszunahme in X-Richtung.

OBJECT.AY Objektnummer,Beschleunigung
Geschwindigkeitszunahme in Y-Richtung.

OBJECT.START (Objektnummer (,Objektnummer,....))
Setzt das Objekt/die Objekte in Bewegung. Ohne Objektnummern werden alle Objekte im derzeit aktiven Output-Window in Bewegung gesetzt.

OBJECT.STOP (Objektnummer (,Objektnummer,...))
Stoppt die Bewegung des Objekts/der Objekte.

OBJECT.ON (Objektnummer (,Objektnummer,...))
Objekt anschalten. Ohne Angabe der Objektnummern werden alle Objekte des aktiven Output-Windows angeschaltet. Wurde mit OBJECT.START eine Bewegung aktiviert, startet sie erst bei Aufruf des OBJECT.ON-Befehls.

OBJECT.OFF (Objektnummer (,Objektnummer,...))
Ausschalten des Objektes/der Objekte.

OBJECT.PRIORITY Objektnummer,Priorität
Dieser Befehl gilt nur für BOBs, nicht für Sprites. Das BOB mit der höheren Priorität (-32768 bis 32767) liegt vor den anderen BOBs.

OBJECT.CLIP (x1,y1)-(x2,y2)
Außerhalb des durch diesen Befehl festgelegten Bereichs werden keine Objekte angezeigt (Betriebssystem-Clipping).

OBJECT.CLOSE (Objektnummer (,Objektnummer,...))
Der Speicher, den das Objekt benötigte, wird durch diesen Befehl wieder freigegeben.

OBJECT.HIT Objektnummer,(MeMask)(,HitMask)
Bestimmt Kollisionsbedingungen für ein Objekt. Default: Alles kann miteinander kollidieren. Durch die 16-Bit-Masken MeMask und Hit-Mask kann bestimmt werden, daß Objekte keine Kollision verursachen.

COLLISION ON/COLLISION OFF/COLLISION STOP
Anschalten, Ausschalten oder Unterbrechen des Kollisions-Event-Trapping.

ON COLLISION GOSUB label
Sofern das Kollisions-Trapping angeschaltet ist, wird bei einer Kollision in die Unterroutine LABEL gesprungen.

GET (x1,y1)-(x2,y2),Arrayname (index(,index))
Grafikbereich in Array ablegen.

PUT (x,y),Arrayname (index(,index...))(,Aktion)
Array in Grafikbereich ablegen, mit Aktionsverb: PSET = Setzen, PRESET = Löschen, AND, OR, XOR = Logische Verknüpfung mit der Hintergrundgrafik, wobei der Defaultwert XOR ist.

SCROLL (x1,y1)-(x2,y2),delta-x,delta-y
Scrollen eines Grafikbereiches, der durch x1 bis y2 festgelegt ist. Delta-x und Delta-y geben an, wieviele Pixel nach rechts oder nach unten gescrollt wird. Links- oder Aufwärtsbewegung durch Angabe negativer Zahlen.

Funktionen

OBJECT.X (Objektnummer)
Gibt die X-Position des Objekts wieder.

OBJECT.Y (Objektnummer)
Gibt die Y-Position des Objekts wieder.

OBJECT.VX (Objektnummer)
Gibt die aktuelle Geschwindigkeit des Objektes in X-Richtung wieder.

OBJEXT.VY (Objektnummer)
Wie oben, aber Y-Geschwindigkeit.

COLLISION (Objektnummer)
ergibt Nummer des Objektes, mit dem das abgefragte Objekt zusammengestoßen ist oder: -1 = obere Window-Grenze, -2 = linke Window-Grenze, -3 = untere Window-Grenze, -4 = rechte Window-Grenze. Bei Objektnummer 0 wird das zuletzt kollidierte Objekt abgefragt, bei Objektnummer -1 erhält man die Nummer des Windows, in dem COLLISION(0) aufgetreten ist.

Stein für Stein

Ein Haus baut man Stein für Stein. Das machen Maurer. Komplexe Programme entwickelt man nach demselben Prinzip und läßt sie dann »zusammenbauen«. Dafür gibt es Linker.

Komplexe Programme sind gerade für Atari ST-Besitzer kein Problem. Speicherplatz ist in Hülle und Fülle vorhanden. Wer umfangreiche Programme entwickelt, weiß aber, daß die Fehlerhäufigkeit proportional zum Umfang ansteigt. Wie begegnet man diesem lästigen Übel? Ganz einfach, man entwickelt ein komplexes Programm in kleinen Teilen. Sie sind überschaubarer, nicht so fehlerintensiv und leichter auszutesten als die großen Ungetüme.

Jeder Atari ST-Benutzer, der nicht nur mit Anwenderprogrammen arbeiten, oder Programme nicht nur im mitgelieferten Basic oder Logo entwickeln will, braucht neben einer anderen Programmiersprache ein Maurer-Programm, einen »Linker« oder »Binder« also. Dank des Linkers kann ein Programmierer ein großes Programm in Teilen entwickeln und sie anschließend zusammensetzen. Programmiersprachen, die auf die Existenz des Linkers bauen, erzeugen nicht ausführbare Objektdateien.

Diese Objektdateien enthalten als linkerlesbaren Zwischencode alle Informationen, die ein Linker braucht, um daraus eventuell mit weiteren Objektdateien oder Routinen aus Bibliotheken ein lauffähiges Programm »zusammenzuhängen«. In Bibliotheken befinden sich, ebenfalls in einem linkerlesbaren Zwischencodeformat, Routinen, die ihre programmübergreifende Nützlichkeit schon bewiesen haben. Das können zum Beispiel Multiplikations- und Divisionsroutinen für reelle Zahlen sein. Der Linker holt sie sich aus der Bibliothek und bindet sie in das zu linkende Programm ein, wenn sie dieses aufruft. Linker können auch Objektdateien und Bibliotheken des gleichen Zwischencodeformats, aber so verschiedenen Ursprungs wie Assembler, C oder Pascal, zu einem lauffähigen Programm vereinen.

Das alles leistet auch der Link68 des Atari-Entwicklungssystems. Er ist auf das Objektcodeformat des Assemblers und des C-Compilers des Entwicklungssystems ausgerichtet. Auch alle anderen Hilfsprogramme im Entwicklungspaket wurden an dieses Objektcodeformat angepaßt. Ein Linker, der diesem Standard folgt, ist »Fastlink« von CCD. Er fand durch das leistungsfähige ST-Pascal eine große Verbreitung.

Der Link68 erzeugt keinen lauffähigen Code. Der Code ist vom Typ »68.K«, den erst das Hilfsprogramm RELMOD in eine lauffähige ».PRG«-Datei umwandelt. Ist Link68 deswegen kein richtiger Linker? Doch, aber er war ursprünglich nicht für den Atari und GEMDOS vorgesehen, sondern für ein anderes Betriebssystem namens CP/M-68K. Beide Betriebssyteme stammen von der Firma Digital Research. CP/M-68K und GEMDOS besitzen Gemeinsamkeiten auf Dateiformatebene, so daß man ohne große Probleme eine CP/M-68K-Datei in eine ».PRG«-Datei umwandeln kann oder auch umgekehrt. Link68 war, wie die meisten anderen Programme, im Entwicklungspaket ein »Überbleibsel« des früheren Entwicklungspakets für CP/M-68K. Fastlink dagegen wurde speziell für GEMDOS entwickelt und erzeugt direkt ausführbare ».PRG«-, ».TOS«- oder ».TPP«-Dateien. Daneben nutzt Fastlink den großen Speicherbereich des Atari aus.

Link68: Ein Linker der 1. Generation

Programme für den Atari ST müssen so geschrieben sein, daß sie überall funktionieren, egal in welchem Teil des Speichers sie liegen. Denn wo sie sich bei der Ausführung befinden, entscheidet nicht mehr der Programmierer, sondern GEMDOS. Durch diese notwendige Positionsunabhängigkeit kann der Speicher mehrere Programme gleichzeitig beinhalten. Das nützt außer den Accessoires noch kein käufliches Programm aus, aber es geht. Damit das Betriebssystem in so einem Fall nicht den Überblick verliert, benötigt es einige »persönliche« Daten des Programms. Eine Aufgabe des Linkers ist es nun, diese Daten, die sich im Programmkopf befinden, zu erzeugen.

Der Programmkopfaufbau entspricht vollständig dem des CP/M-68K. Nur die Bedeutung der Felder hat sich teilweise geändert. Die Zahl 601A(hex), mit der jedes Programm anfangen muß, gibt an, daß das Programm positionsunabhängig ist, und daß das eigentliche Programm aus einem zusammenhängenden Codebereich, Daten und Variablenblock besteht. Unter CP/M-68K konnte dieses Feld auch einen anderen Wert annehmen. Die Felder von 12 bis 1A(hex) werden von GEMDOS noch nicht benutzt, müssen aber die Zahl 0 enthalten.

Die Zahlen in den Feldern 2 bis 0A (hex) geben zusammen die Größe des lauffähigen Programms an. Ein Programm besteht nach Konversion aus drei verschiedenen Segmentarten. Dem eigentlichen Programmcodeteil folgt der Bereich der initialisierten, darauf der Bereich der nicht initialisierten Daten. Letzterer befindet sich nicht auf der Programmdatei und ist ein Bereich, der beim Laden für das Programm reserviert und auf 0 gesetzt wird. Neben diesen Teilen besitzt jede Programmdatei auch noch einen Bereich mit Relozierungsinformation, in dem steht, wie das Programm abhängig von dem Ort geändert werden muß. Aus diesen fünf Teilen, Programmkopf, Codesegment, den beiden verschiedenen Datensegmenten und dem Relozierungsblock setzt sich jede Programmdatei zusammen. Zwischen dem Codeteil und dem Relozierungsblock kann aber auch noch eine Symboltabelle liegen. Sie ist für die eigentliche Funktion des Programms bedeutungslos, und ob sie vorhanden ist oder nicht, hängt davon ab, mit welchen Steuerschaltern der Linker aufgerufen wurde. Die Länge der Symboltabelle steht im Feld 0E (hex) des Programmkopfes. Die Tabelle selbst enthält die symbolischen Namen der Quellsprache.

Der Programmkopf jedes Programms für den Atari ist folgendermaßen aufgebaut:

Entfernung vom Programmanfang in Byte	Bedeutung des Werts	Segmenttyp
00hex	Anzahl der Bytes im Code-Teil	Text
06hex	Anzahl der Bytes im initialisierten Datenteil	Data
0Ahex	Anzahl der Bytes im nichtinitialisierten Datenteil	BSS
0Ehex	Anzahl der Bytes der Symboltabelle	BSS
12hex	Reserviert, muß 0 sein	BSS
16hex	Reserviert, muß 0 sein	BSS
1A bis 1Bhex	Reserviert, muß 0 sein	BSS

Faßt man alles Bisherige zusammen, so ist jede Programmdatei folgendermaßen gegliedert:

Programmkopf
Code- und Datensegmente
Symboltabelle (wenn vorhanden)
Relozierungsinformation

Für den Linker ist die Erzeugung und Verarbeitung der Symboltabelle die Hauptaufgabe, und die Übergabe an den Debugger nur eine Nebensache. Sie beinhaltet, wie erwähnt, alle symbolischen Namen eines Programms oder Programmteils. Mit jedem Namenseintrag, der bis zu sieben Buchstaben lang sein darf, ist ein Feld verknüpft, das die Bedeutung oder den Wert des Namens angibt, und ein Feld, das den Namenstypus beschreibt. Jeder Eintrag in der Symboltabelle hat dann folgenden Aufbau:

Name 0..6
0 7
Typ 8..9
Wert A..D

Ist der Name kürzer als sieben Buchstaben, wird der Platz mit Nullen aufgefüllt. Der Typ eines Namens kann eine oder mehrere der folgenden Bedeutungen annehmen:

0100(hex): zum uninitialisierten Datensegment gehörend
0200(hex): zum Codesegment gehörend
0400(hex): zum initialisierten Datensegment gehörend
0800(hex): externer Bezug
1000(hex): entspricht Register
2000(hex): global bekannt gegeben
4000(hex): ist gleich
8000(hex): definiert

Eine Zahl A400(hex) im Typenfeld bedeutet also, daß der Name definiert ist, zum Datenbereich gehört und die Eigenschaft besitzt, im Programm glo-

bal bekannt zu sein. Werden Namen nicht ausdrücklich nach außen hin bekanntgegeben, so weiß der Linker, daß sie nur innerhalb der Objektdatei Geltung besitzen, in der sie vorkommen. Benutzt man undefinierte Namen in einem Programm, so muß man dem Linker mitteilen, daß sie außerhalb dieses Programms definiert sind. Das kann entweder in einer anderen Objektdatei geschehen sein, die man dann anhängt, damit der Linker »die Referenz auflösen« kann, oder der Name befindet sich in einer Bibliothek. Ein kleines Assemblerbeispiel mit der dazugehörigen Symboltabelle verdeutlicht das:

```
Beispiel TEST.S
* Was steckt in einem Namen?
wert  = 12    ; wert ist 12
  .xref MENU_REG    ; importiere
                      diesen Namen
  .XDEF heap,heap0  ; exportiere
                      diese Variablen
  .globl dbugflag   ; exportiere
                      diese Variable
  .text
  jsr MENU_REG
 testl: clr.w -(a7) ;
lokale Marke
  trap #1
  .bss              ;
nicht initialisierter Daten-
bereich.
 heap:  ds.l 0
 heap0: ds.l 0
  .data             ;
initialisierter Datenbereich
 dbugflag:ds.w wert
```

Nach der Assemblierung wird NM68 TEST.O eingetippt: MENU__REG 0

```
global external abs
 wert        C equ abs
 heap      0 global bss
 heap0     0 global bss
 dbugflag  0 global data
 testl       6 text
```

Das Feld, in dem die Zahlen stehen, ist der Wert des Namens. Der Name »testl« ist nur innerhalb des Programms »test.s« bekannt und gehört zum Codesegment. Der Wert von testl ist die Adresse des Namens im Programm relativ vom entsprechenden Segmenttyp ab gerechnet. »Heap«, »heap0« und »debugflag« sind Namen, die global bekannt sind. Wert ist ein Name, der – wo immer er vorkommt – durch seinen absoluten (abs) Wert ersetzt wird. Ebenso wird MENU__REG überall in test.s durch seinen Wert ersetzt. Er ist aber erst in einer anderen Objektdatei oder Bibliothek ausfindig zu machen. Das zeigt der Typus »external« im Namenseintrag.

Link68 und Fastlink können Bibliotheken nach Namen durchsuchen. Bibliotheken, Routinensammlungen also, lassen sich mit dem Programm AR68 aus dem Entwicklungssystem entwickeln und einsehen. Bibliotheken *gliedern sich in sogenannte Module, in* denen sich verwandte Routinen befinden sollen. Sehen wir uns einmal die Module der Bibliothek PASLIB an:

```
AR68 TV PASLIB
 rw-rw-rw- 0/0  952 STRUKT.O
 rw-rw-rw- 0/0  2648 CMDLINE.O
 rw-rw-rw- 0/0  3128 READREAL.O
 rw-rw-rw- 0/0  1456 SET.O
  :: ::
```

Der Zeichenkette »rw-...« fällt unter GEMDOS keinerlei Bedeutung zu. Die folgende Zahl gibt die Größe des Moduls an. Anschließend folgt der Modulname. Mit dem Verwaltungsprogramm AR68 können Sie auch einzelne Module extrahieren.

Bibliotheken nicht nur für Bücher

Das Kommando »AR68 XV PASLIB READREAL.O« sollte eigentlich die Datei READREAL.O aus PASLIB extrahieren und unter dem Namen READREAL.O auf die Diskette schreiben. Aber AR68 gibt eine Fehlermeldung aus, nach der er READREAL.O nicht in der Bibliothek findet. Es handelt sich hierbei um einen Fehler im Archiver. Er kann nämlich keine Namen in Großbuchstaben herausfinden. Man lädt die PASLIB in den Debugger, sucht den Namen READREAL.O und wandelt ihn in Kleinbuchstaben um. Daraufhin gibt man das Kommando noch einmal ein. Nun befindet sich die Objektdatei READREAL.O auf der Diskette, und man kann sich die in ihr benutzten Namen und Routinen wieder mit NM68 ansehen:

```
NM68 readreal.o
readreal   C global text
IO_CLEAR   0 external abs
float      0 external abs
get        0 external abs
setin0     0 external abs
realmult   0 external abs
realadd    0 external abs
pwrof10    0 external abs
float1     0 external abs
realdiv    0 external abs
IOERR1     0 external abs
realneg    0 external abs
```

Man erkennt an dem nur einmal vorkommenden Typ »global«, daß im Modul READREAL nur eine Routine definiert ist. Diese Routine readreal benutzt eine Menge nicht in ihr selbst definierter Namen.

Findet der Linker einen gesuchten Namen in einem Modul einer Bibliothek, so bindet er gleich das ganze Modul in das entstehende Hauptprogramm ein. Kommen in dem Modul wieder nicht aufgelöste Referenzen vor, wie im Falle von READREAL.O, so kann das zu einer Kaskade von weiteren Moduleinbindungen führen. Wichtig bei der Anordnung der Module in der Bibliothek ist, daß Module keine Symbole benutzen, die »früher« in der Bibliothek definiert wurden; der Linker sucht die Bibliothek nur in einer Richtung, vom Anfang zum Ende hin, ab.

Nun wissen Sie genug, um die Funktion der Steuerungsschalter von Fastlink zu verstehen. Wird Fastlink ohne Argumente aufgerufen, erscheint die Syntax der Kommandozeile und die Vorgaben, die er versteht:

```
Fastlink
ST LINK V1.0 -
fast TOS linker by Jörg Lohse,
Copyright  1986 by J. Lohse
 Usage: LINK {-<options>}
[<file>=][@]<file>{,
[@]<file>}
   @<file>  insert <file>
into command line
 Options: u   ignore undefined
symbols
  s   enter symbols into output
file
  m[=<file>]  write map-
file, default: stdout
  r[=<address>] raw output
file, default: FA0000
```

Enthält die Kommandozeile ein »@« vor einem Dateinamen, so betrachtet Fastlink diese Datei als erweiterte Steuerzeile. Die Vorgabe »-u« dient dazu, trotz undefinierter Symbole ein lauffähiges Programm zu erhalten. Wird in einem C-Programm beispielsweise die Ausgaberoutine »printf« benutzt, müßte man auch das Modul miteinbinden, das reelle Zahlen behandelt, weil »printf« allgemein die Ausgaben formatiert. Verwendet aber dieses Programm keine reellen Zahlen, so kann der Verweis auf die entsprechenden Routinen unaufgelöst bleiben. Das verkürzt das Programm. Der Befehl »-s« erzeugt die Symboltabelle für die ».PRG«-Datei, »-m« gibt die Symboltabelle mit der jetzt bekannten endgültigen Adresse aus und »-r« erzeugt das Bild eines ausführbaren Programms ohne Programmkopf und Relozierungsinformation für eine feste Adresse. Die voreingestellte Adresse ist $FA0000, was auch auf den Zweck dieses Befehls hinweist: Diese Adresse ist der Anfang des EPROM-Bereichs des Atari und man kann mit diesem Schalter Programme für diesen Bereich produzieren.

(Claus Brillowski/hb)



Fortan nur noch Fortran

Fortran ist aus Mathematik und Wissenschaft nicht mehr wegzudenken. Auf Computern der 16-Bit-Generation erlebt diese altbewährte Programmiersprache einen neuen Frühling.

Fortran war bisher allein größeren Rechenanlagen vorbehalten. Obwohl gerade in den letzten Jahren revolutionäre Fortschritte bei der Entwicklung von höheren Programmiersprachen erreicht wurden und Fortran viele leistungsfähige Konkurrenten bekommen hat, erfreut sich diese Programmiersprache noch immer großer Beliebtheit. Dafür gibt es viele gute Gründe.

Zum einen besitzt Fortran für die Verarbeitung von Formeln und allgemeinen mathematischen Problemen eine programmierfreundliche Struktur. Beispielsweise erlaubt sie spezielle Unterprogramme, sogenannte »Functions«, die die Definition von Funktionsgleichungen zulassen. Das folgende einfache Beispiel zeigt die Definition einer Funktion G(x) im Function-Unterprogramm. An jeder Stelle des Programms ist die Funktion dann aufrufbar, wobei sich xbeliebige Werte, Variablen und Terme zuordnen lassen:

```
C = G(4) + 3 / G(L+7)
.
.
FUNCTION G(x)
REAL x
G(x) = 2 * x + 3
RETURN
END
```

Fortran rechnet wahlweise mit einfacher oder doppelter Genauigkeit. Dabei hat man die Wahl zwischen acht oder 16 Nachkommastellen.

Eine Besonderheit stellt die Verarbeitung komplexer Zahlen dar. Diese werden aufgeteilt als Real- und Imaginärteil abgelegt. Der Benutzer braucht bei der Zuordnung der Werte keine Besonderheiten zu beachten.

Selbstverständlich erlaubt Fortran auch die Programmierung von Feldern. Felder lassen sich jederzeit neu dimensionieren und umdefinieren. Ohne Datenverlust wird beispielsweise ein 2x3-Feld in ein 5x4-Feld umgewandelt. Jedes Feld besitzt Elemente des gleichen Datentyps:

```
DIMENSION  A(3,6,32)
DIMENSION  I(2,8)
```

In diesem Beispiel ist A ein dreidimensionales Feld, das Fließkommazahlen mit einfacher Genauigkeit aufnehmen kann, I ein zweidimensionales Feld, das nur ganzzahlige Werte beinhaltet. Fortran besitzt eine implizite Typenvereinbarung. Das heißt, alle Variablen und Feldnamen mit den Anfangsbuchstaben A bis H und O bis Z sind automatisch Real-Zahlen, beziehungsweise alle diejenigen von I bis N Integer-Zahlen. Dieses gilt solange, bis der Programmierer eine explizite Typenanweisung vornimmt.

Eine weitere Hilfe bei der Verarbeitung mathematischer Ausdrücke leistet die umfangreiche Standardbibliothek. Sie enthält nützliche Befehle wie zum Beispiel:

NINT(a):	nächster ganzzahliger Wert
MOD(a/b):	Modulo (Divisionsrest)
MAX(a,b), MIN(a,b):	geben den größten beziehungsweise kleinsten Wert aus der Menge (a,b)
CONJG(c):	berechnet das konjungiert Komplexe
DIM(a,b):	positive Differenz zweier Zahlen
REAL(c):	Realteil einer komplexen Zahl
LOG10(a):	dekadischer Logarithmus
TANH(a):	Hyperbeltangens

und so weiter.

Das sind nur einige wenige von über einhundert weiteren Befehlen. Der Umfang einer solchen Standardbibliothek hängt von der Version und dem Anbieter ab.

Großer Softwarepool

Die modulare Aufbauweise von Fortran unterstützt den Aufbau und die Anwendung von Bibliotheken und Standardprozeduren. Das spart Programmierzeit und Arbeitsaufwand. Obwohl hier nur eine Auswahl der wichtigsten Eigenschaften von Fortran zur Sprache kamen, ist bereits klar zu erkennen, welchen Vorteil Fortran gegenüber anderen Programmiersprachen auf dem Gebiet der Darstellung und Verarbeitung von mathematischen Problemen hat.

Was Fortran weiterhin gerade für Studenten und Ingenieure so interessant macht, ist das große Angebot an Software. Nahezu alle Universitäten und technisch-wissenschaftlichen Einrichtungen arbeiten mit Fortran. Daher verwundert es nicht weiter, daß viele leistungsfähige und anspruchsvolle Programme in dieser Sprache geschrieben sind. Da Fortran für viele Studenten und Wissenschaftler zu den Pflichtübungen zählt, existiert außerdem ein großes Heer von Programmierern. Deshalb wird auch in absehbarer Zeit Fortran, selbst gegenüber leistungsfähigen Alternativen, in Wissenschaft und Technik seine Stellung halten.

Einen weiteren klaren Vorteil gegenüber anderen Programmiersprachen verdankt Fortran seiner Standardisierung. 1966 wurde vom American National Standards Institut (ANSI) der erste Standard festgelegt (Fortran IV), der letzte kam 1977 für das Fortran 77 heraus. Diese Normung verhindert die Entstehung der vielen Dialekte, wie sie bei anderen Sprachen immer wieder auftauchen, weitgehend. Programme werden somit nahezu problemlos auf verschiedene Systeme übertragen.

Doch wo viel Licht ist, da ist auch Schatten: Eine Schwäche zeigt Fortran bei der Verarbeitung von Zeichenketten. Es gibt zwar einige Kommandos wie LEN (Feststellen der Länge einer Zeichenkette), INDEX (Suchen einer Zeichenfolge in einer Zeichenkette) und LGE/LGT/LLE/LLT (lexikalische Vergleiche von Zeichenketten). Dennoch bleibt die Zeichenverarbeitung eine recht mühsame Angelegenheit. Sehr gewöhnungsbedürftig verhalten sich auch die Ein- und Ausgabeprozeduren:

```
    WRITE (10,100) 3,'mal Hallo'
100 FORMAT(2X,I2,2X,A5)
```

Erst durch die Angabe einer Formatzeile wird die Form der Ausgabe oder Eingabe festgelegt, während der Befehl WRITE das Peripheriegerät und die angesprochene Formatzeile bestimmt. Die Eingabe erfolgt ähnlich der Ausgabe:

```
   READ (5,20) A
20 FORMAT(F7.4)
```

Fortran ist nach wie vor eine sehr leistungsfähige Compiler-Sprache. Mit dem vermehrten Einsatz von Fortran auf Computern der 16-Bit-Klasse darf man in Zukunft sicherlich mit mehr Programmen rechnen. (Christian Träger/ Michael Zwenger/Matthias Rosin/hb)

Heißer Draht zu neuen Tips

DFÜ-Freaks ist sie schon seit langer Zeit ein Begriff: BIX, eine der großen Mailboxen in den USA. Einige sehr interessante Rubriken beschäftigen sich besonders mit den 68000-Computern Atari St und Amiga.

Eine nützliche Einrichtung in der Super-Mailbox BIX sind die »Bulletin Boards«, Sammelstelle für Fragen. Wir haben uns eingelogt und nach Tips und Tricks zum Atari ST und Amiga gewühlt. Das Besondere an BIX ist, daß die Antworten im Bulletin Board zum Teil von den Entwicklern der Computer selbst stammen. Zum Teil auch beantworten gar die Größen aus der Softwareindustrie höchstpersönlich Fragen zu ihren Programmen. Das garantiert natürlich ständige Aktualität und die Sicherheit, daß es kaum eine Frage gibt, auf die nicht jemand in irgendeiner Form antwortet.

Fragen und Antworten zum Amiga

Immer wieder tauchen Fragen zur Grafikauflösung und zu den Monitoren des Amiga auf.

»Ich bin auf der Suche nach einem Monitor für meinen Amiga. Ich habe aber noch kein Modell gesehen, das genau die Auflösung von 640 mal 400 Pixeln darstellen kann. Ein Modell, das in Frage käme, hat aber eine Auflösung von 720 mal 480 Pixeln. Kann ich diesen Monitor nun verwenden?«

Die Antwort darauf gibt Dale Luck, der Entwickler des Grafikteils im Amiga:

»Die 640 beziehungsweise 720 Pixel breite Auflösung in der Horizontalen ist fast identisch. Ein 720-Pixel Monitor bringt seine Punkte in einer längeren Linie unter, die Videobandbreite ist aber gleich. Diejenigen, die ihre eigenen Grafikroutinen verwenden (keine Einsprünge ins Intuition), werden festgestellt haben, daß der Amiga in der Lage ist, bis zu 700 mal 220 Pixel darzustellen. Der sichtbare Teil unterscheidet sich jedoch von Monitor zu Monitor.«

Ein weiteres, oft diskutiertes Thema stellt der Anschluß von 5¹/₄-Zoll-Laufwerken an den Amiga dar. Hierbei ergeben sich zwei Probleme.

Problem 1: Der Steckverbinder an der Floppy ist kein DB-25, sondern ein DB-23. Solch einen Stecker kann sich jeder ganz einfach selbst »bauen«, indem er bei einem DB-25 die letzten zwei Pins abkneift. Danach ändern nur noch die Pins 14 bis 24 ihre Bezeichnung in Pin 13 bis 23, und die Sache funktioniert.

Auf ein zweites Problem stößt man bei der Steckerbelegung. Dort finden sich zwei Signale, die nicht mit der Standardbelegung übereinstimmen: Pin 10 und Pin 11 am Amiga.

Aufklärung bietet die vollständige Belegung des Floppysteckers im Amiga:

Pin 1	Ready-Signal	Pin 14	Schreibschutz
Pin 2	Read Data	Pin 15	Spur 0
Pin 3-7	Masse	Pin 16	Write Gate
Pin 8	Motor	Pin 17	Write Data
Pin 9	Drive Select C	Pin 18	Step
Pin 10	Motor Clear	Pin 19	Direction Select
Pin 11	Disk Change (zeigt an, daß Diskette gewechselt wurde)	Pin 20	Drive Select D
		Pin 21	Drive Select B
		Pin 22	Index-Signal
		Pin 23	+12 Volt (reicht nur für ein 5¹/₄-Zoll-Laufwerk)
Pin 12	+5 Volt		
Pin 13	Seitenauswahl		

Die Verbindung zur Außenwelt macht oft Schwierigkeiten. Die Joystickports des Amiga bieten hier verschiedene Möglichkeiten. Eine der meistgestellten Fragen betrifft die Funktion der beiden Eingänge »PotX« und »PotY«.

»Darf man der Bezeichnung dieser zwei Eingänge entnehmen, daß sich Analog-Joysticks, wie sie vom IBM oder Apple her bekannt sind, anschließen lassen? Wie schnell lassen sich diese Eingänge lesen?«

Jez San, Programmierer bei Argonaut Software, klärt diese Frage. »Die A/D-Wandler im Amiga eignen sich in der Tat für Analog-Joysticks. Sie messen einen Widerstand nach folgender Methode: Ein Kondensator wird jedesmal ein bißchen aufgeladen, wenn der vertikale Synchronisationsimpuls auftritt. Und die Zeit, die nötig ist, um den Kondensator voll aufzuladen, ist ein Maß für den Widerstand an PotX oder PotY.«

Fragen und Anworten zum Atari ST

Bekannterweise besitzt der Atari ST drei verschiedene Auflösungen: Die hohe Auflösung läßt sich nur auf dem speziellen Atari-Monitor erzielen. Interessant sind die beiden anderen Auflösungen, die mittlere mit 640 x 200 Pixel bei vier Farben und die geringe mit 320 x 200 Pixel bei 16 Farben gleichzeitig.

Einige Programme setzen die Umschaltung zwischen dem Multicolor-Bildschirm und dem 80-Zeichen-Bildschirm voraus. Aber es gibt Probleme, will man die Umschaltung ohne ein erneutes Booten des Systems vornehmen. Auch hier weiß Jez San von Argonaut Software Rat.

»Es ist für ein Programm ganz einfach, zwischen diesen zwei Modi hin- und herzuschalten. Alles, was man tun muß, ist, das Register »shiftmd« (Video Shift Mode) schnell genug zu manipulieren. Diese einfache Methode benutzen wir in unseren Spielen. Das klappt allerdings nicht mehr, will man GEM benutzen oder dorthin zurückkehren. GEM wurde nämlich nicht dafür vorgesehen, in verschiedenen Grafik-Modi zu arbeiten. Es gibt aber bestimmt irgendwelche Programmiertricks, die solche Schwierigkeiten umgehen. Aber auf geradem Wege sind mir keine Lösungen bekannt.«

Auch die Behandlung der I/O-Ports stellt ein beliebtes Betätigungsfeld dar. Dazu Steve Krenek: »Kann der Atari während des Zugriff auf einen I/O-Port noch andere Dinge erledigen, wie beispielsweise der Amiga? Oder muß er warten, bis der Zugriff auf dieses Port abgeschlossen ist?«

Jim Tittsler von Atari nahm sich der Frage an. Während des Zugriffs auf die Diskette kann der Atari seiner Auskunft nach nebenbei noch andere Sachen erledigen. Das liegt daran, daß die ankommenden Daten zwischengespeichert werden. So erfolgt nur dann ein Zugriff

auf den Prozessorbus, wenn der interne FIFO-Speicher (First In First Out) geleert werden muß. Umsonst bleibt der Prozessor frei für andere Aufgaben.

Auch beim Atari stellt die Programmierung der Schnittstellen manchen Benutzer vor Probleme, und zwar mit den Joystickports. Entsprechen sie denen des C64 und lassen sie sich bidirektional nutzen?

Jim Tittsler kann auch hier helfen: »Die Ports besitzen die gleiche Belegung wie die des C 64 oder der kleinen Ataris. Es fehlen lediglich die Analog-Eingänge. Ein weiterer Unterschied liegt darin, daß man von diesen Ports nur lesen kann. Der Grund ist darin zu suchen, daß diese Ports vom Tastaturprozessor abgefragt werden. In seinem Programm ist aber keine Möglichkeit vorgesehen, diese Ports als Ausgänge zu schalten. Allerdings kann man auf den Druckerport als Ausgang ausweichen, und eine Reihe anderer Ports, die frei programmierbar sind. Ein Bit ist an der Videobuchse abgreifbar, ein anderes Bit wartet am Soundchip auf »Sonderaufgaben«. Dem Bastler stehen damit eine ganze Menge an Möglichkeiten zur Verfügung.«

Jez San hat einen Weg gefunden, auf einer einseitigen Diskette 409 KByte an Daten unterzubringen. Er löste das Problem mit Hilfe von 1-KByte-Sektoren statt den üblichen 512-Byte-Sektoren. Seine Frage lautet nun, warum Atari auf die zusätzlichen 50-KByte-Speicher verzichtet und 512-Byte-Sektoren verwendet.

Darauf antwortet noch einmal Jim Tittsler, ein eifriger Benutzer von BIX. »Es gibt wirklich ein paar Leute, die zehn statt neun Sektoren nutzen. Aus ein paar Gründen haben wir uns für die 512-Byte-Sektoren entschieden, einer der wichtigsten war die Kompatibilitätsfrage. Aber wie man sieht, bereitet es dem Atari keine Schwierigkeiten, eine Vielzahl von Formaten zu unterstützen.«

Bei Benutzung einer solchen Diskette können allerdings zwei Probleme auftauchen: Zum einen ergeben sich Schwierigkeiten beim Kopieren dieser Diskette. Viele Kopierprogramme sind nur in der Lage, Disketten zu kopieren, die mit neun Sektoren pro Spur formatiert wurden. Dazu kommt noch, daß nur bis Spur 79 kopiert wird. Ein Teil der Information befindet sich aber auf den Spuren 80, 81 und 82 und geht somit verloren. Zum zweiten sind einige Laufwerke nicht in der Lage, Spur 82 anzufahren.

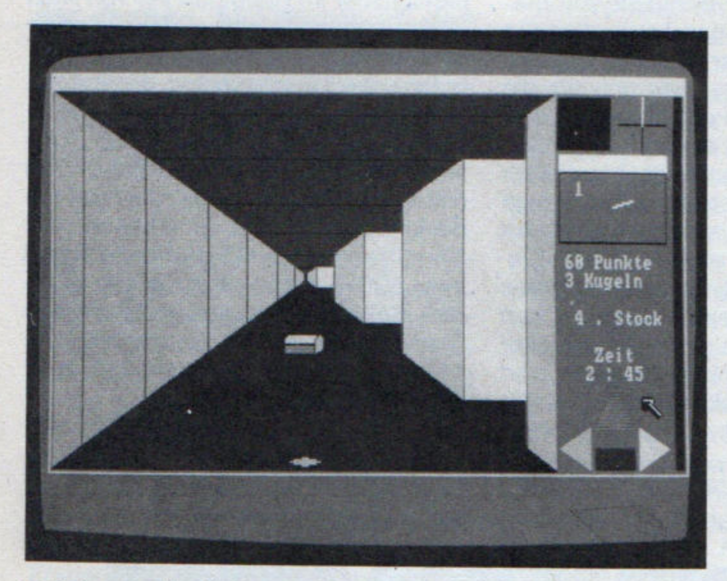
Zum Abschluß unserer Fragestunde nun noch ein Bonbon für alle DFÜ- und Atari-Freunde:

Atari sponsert eine Mailbox in Sunnyvale, dem Sitz von Atari. In einem kleinen Zimmer im Hauptquartier von Atari gehen wöchentlich bis zu 1000 Anrufe ein. Dort hagelt es ständig Neuigkeiten, technische Unterstützung, Listen von Usergroups und Software zum »Downloaden« für alle Atari-Computer. Das Mailboxsystem heißt »Atari Base« und hält vier Leitungen bereit, die rund um die Uhr besetzt sind. Es arbeiten zwei STs mit je einer 20-MByte-Harddisk und zwei 800XL mit je einer Corvus-Harddisk. Seit letztem August bedienen sich Atari-Besitzer dieser Einrichtung. Die Atari Base ist jeden Tag unter 001/408/7455308 zu erreichen.

(Udo Reetz/lg)

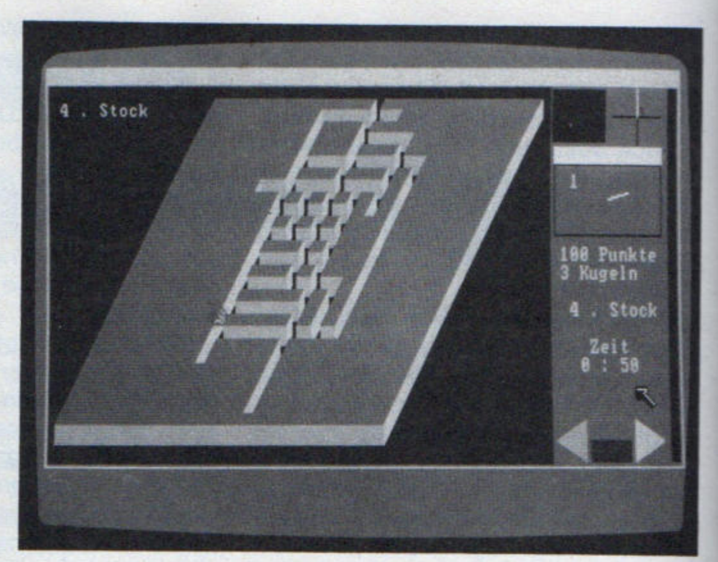
Listiges Labyrinth

Sie stehen vor einer fast unlösbaren Aufgabe. Unter Zeitdruck müssen Sie in einem vierstöckigen Irrgarten vier Ringe finden. Steigen Sie ein in die 3D-Welt des Amiga!



Geheimnisvolle Kisten und Zauberkugeln

Als geübter Schatzsucher begeben Sie sich dorthin, wo die größte Gefahr droht. Diesmal wollen Sie unbedingt die vier goldenen Schlüssel aus einem Labyrinth entführen, das gewalttätige Gnome bewachen. Aber damit nicht genug. Ihre Zeit ist knapp und die Nahrung ebenso. Zum Glück liegen im Labyrinth noch einige Kisten herum, wahrscheinlich Relikte Ihrer glücklosen Vorgänger. Und wie es im Reich der Elfen, Zauberer und Gnome eben ist, enthalten die Kisten nicht nur Gold, Nahrung oder Zauberkugeln. Einige sind auch mit Zaubersprüchen behaftet, die Sie an andere Orte bringen oder die Ihnen etwas von der verlorenen Zeit zurückgeben. Mit den Zauberkugeln hat es etwas Besonderes auf sich. Der Besitzer dieser Kugeln kann es nämlich frohgemut mit den Gnomen aufnehmen. Er erhält sogar noch 100 Punkte und 50 Nahrungsrationen dazu. Aber wehe dem, der mit einem Gnom kämpft, ohne eine Kugel bei sich zu führen. Demjenigen werden 100 Punkte und 50 Nahrungsrationen abgezogen. Über einen niedrigen Score kann man erhaben hinwegsehen, aber Nahrungsmangel macht sich schnell und unangenehm bemerkbar. Die Beine werden schwer und versagen ihren Dienst, und die Müdigkeit droht den Kämpfer zu übermannen. Daher sollte man stets auf den schwarzen Balken am rechten Bildrand achten, der die Anzahl der Nahrungsrationen angibt. Bedeutungsvoll sind auch die Geräusche im Labyrinth, da die Tonhöhe die Nähe zu einem Ring verrät. Je höher der Ton, desto näher der Ring. Als Hilfe zeigt ein Richtungszeiger den Weg – leider nur Luftlinie – und die Entfernung zu einem Ring. Als geübter Abenteurer führen Sie natürlich noch andere Hilfsmittel mit sich. Am oberen rechten Rand befindet sich ein Kompaß, der Ihren Aufenthaltsort im Labyrinth und Ihre momentane Blickrich-



»Maze« – Abenteuer im 3D-Labyrinth

tung wiedergibt. Die Zahl unter dem gelben Kästchen gibt das Stockwerk an, in dem der Ring versteckt ist. Durch Anklicken des Kästchens schalten Sie auf einen anderen Ring um. Die drei Pfeile am unteren rechten Rand dienen als Steuerung. Ob 90 Grad nach rechts oder links oder in die angezeigte Richtung, ein Druck auf den linken Mausknopf genügt. Im gesamten Spiel kommt nämlich nur die Maus zum Zuge. Das Rechteck zwischen den Richtungspfeilen hat eine besondere Bedeutung. Wenn Sie es anklicken, erscheint eine Übersichtskarte des Stockwerks, in dem Sie sich gerade aufhalten. Das kann sehr dienlich sein, wenn man den Aufzug oder den Ausgang sucht, es kostet aber auch wertvolle Zeit. Durch einen weiteren Druck auf den linken Mausknopf kehren Sie in das Spiel zurück. Das Bild des Labyrinths ist dreidimensional und verändert sich mit Ihrer Blickrichtung. Als optische Hilfe hinterlassen Sie Punkte, um festzustellen, welche Orte Sie bei Ihrer Suche schon abgeklappert haben. Ariadne läßt grüßen! Nachdem Sie ein Stockwerk vollständig durchsucht haben, bringt Sie der Aufzug in eine andere Ebene. Die Lifte befinden sich übrigens immer im Norden. Obwohl Sie zu den größten Helden Ihres Landes gehören, sollten Sie nicht versuchen, durch die Wände zu gehen. Der Mißerfolg könnte Ihrem Selbstbewußtsein schaden. Und Selbstbewußtsein ist vonnöten, wenn Sie diese Aufgabe in nur 20 Minuten bewältigen wollen.

»Maze« ist in AmigaBasic geschrieben und läuft nur mit 512 KByte Speicher. Wenn Sie »Maze« vom CLI-Modus aus starten, schließen Sie alle unnötigen Fenster. Mit dem Programm »Preferences« stellen Sie bitte die Textdarstellung auf 60 Zeichen pro Zeile. Zuerst wird das Labyrinth berechnet, was einen kleinen Augenblick dauert. Im Spiel selbst beweist der Amiga, was in ihm steckt. Obwohl »Maze« in Basic geschrieben und das Listing sehr kurz ist, merkt man beim Spielen, dank der komplexen Handlung und dem schnellen Bildaufbau, nichts davon. (Thomas Rupp/gn)

Steckbrief	
Name:	Maze
Computer:	Amiga 512 KByte
Checksummer:	–
Datenträger:	Diskette

```
'       AMIGA (512 KByte)
'          >>MAZE<<
'        Thomas  Rapp
'       60-Zeichen-Modus
'
 CLEAR ,32000,4000 : DEFINT a-z  : RANDOMIZE TIMER
 DIM xp(2,7,4) : DIM yp(2,7,4) : DIM b(500) : DIM man (150)
 DIM ys(7) : DIM r(27,27,4) : DIM p(27,27,4)
 sch=2 : ' Schwierigkeitsstufe
 DATA 0,11,34,57,71,82,88,91
 DATA 0,-1,-1,0,1,0,1,0,0,-1,0,1,0,1,1,0,-1,0,-1,0,0,1,0,-1
 FOR v=0 TO 7 : READ y : ys(v)=y  : NEXT v
 FOR v=1 TO 4
 READ x,y : rx(v)=x  : ry(v)=y
 READ x,y : rxl(v)=x  : ryl(v)=y : READ x,y : rxr(v)=x  : ryr(v)=y
 NEXT
 SCREEN 1,640,200,3,2  :  SCREEN 2,640,200,3,2
 WINDOW 1," ", ,16,1 : GOSUB farbe
 WINDOW 2," ", ,16,2 : GOSUB farbe
 LINE (8,1)-(18,2),3,bf : LINE (10,2)-(16,4),6,bf
 LINE (12,5)-(14,5),3   : LINE (10,6)-(16,10),3,bf
 LINE (12,11)-(6,16),5  : LINE (14,11)-(20,16),5
 LINE (11,7)-(1,10),5   : LINE (15,7)-(25,5),5
 PSET (11,3),2 : PSET (14,3),2 : GET (1,1)-(25,16),man
 LINE (0,0)-(60,20),4,bf
 CIRCLE (41,4),5,0,.4,3.14,.8 : CIRCLE (11,4),5,0,1.4,3.14,.8
 LINE (6,5)-(36,10),0,b : LINE (13,0)-(39,0),0
 LINE (36,10)-(45,8),0 : LINE -(45,3),0
 PAINT (26,3),5,0 : PAINT (41,6),6,0 : PAINT (26,8),7,0
 PAINT (50,15),0  : GET (6,0)-(45,10),b
 LINE (0,0)-(630,185),3,bf : COLOR 4,3
 FOR x=0 TO 630 STEP 11 : LINE(x,0)-(x,185),4 : NEXT
 FOR y=0 TO 185 STEP 5 : LINE(0,y)-(630,y),4 : NEXT
 LOCATE 7,29:PRINT " MAZE ":LOCATE 14,22:PRINT " Irrgartenberechnung "
 FOR z=1 TO 4:FOR x=1 TO 27:FOR y=1 TO 27:r(x,y,z)=1:NEXT y,x,z
  x1=INT(RND*18+4)
   FOR z=1 TO 4
   LOCATE 17,27 : PRINT z". Stock "
again4: y1=1 : r(x1,2,z)=0 : r(x1,1,z)=0 : xz=x1 : yz=2 : u=0 : xe=x1
again3: u=u+1
again2: x=INT(RND*3-1) : y=INT(RND*3-1)
        IF ABS(x)=1 AND ABS(y)=1 OR x=0 AND y=0 THEN again2
        st=INT(RND*2)*2+4
        IF x=-1 THEN IF xz-st <  2 THEN x= 1 : GOTO ok
        IF x= 1 THEN IF xz+st > 26 THEN x=-1 : GOTO ok
        IF y=-1 THEN IF yz-st <  2 THEN y= 1 : GOTO ok
ok:  FOR o=1 TO st
  xz=xz+x : yz=yz+y
  IF z=1 THEN un=27 ELSE un=26
  IF yz=>un THEN r(xz,yz,z)=0:IF z=1 THEN xb=xz:yb=yz:GOTO fertig ELSE GOTO fert
ig
  r(xz,yz,z)=0
  NEXT o
     GOTO again3
fertig: IF u>180 OR u<28 THEN GOSUB loeschen : GOTO again4
      NEXT z
  FOR b=1 TO 7
   x=ys(b)/.3672: x(1,b)=x  : y(1,b)=ys(b) : x=ys(b-1)/.3672 : x(2,b)=x
   y(2,b)=ys(b) : x=ys(b-1)/.3672 : y=ys(b-1) : x(0,b)=x  :  y(0,b)=y
  NEXT b
  FOR v=1 TO 4
wert1: x=RND*18+4 : y=RND*18+4
```

Listing »Maze«, Abenteuer im Labyrinth

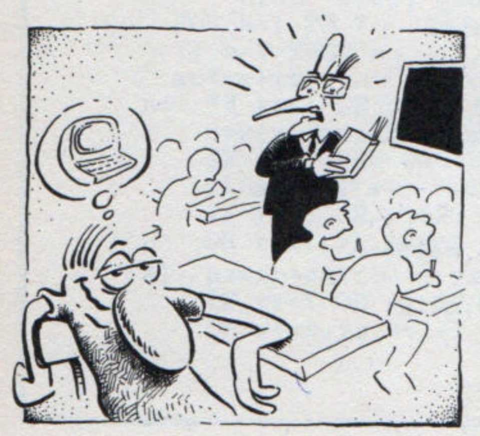
```
   IF r(x,y,v)=0 THEN p(x,y,v)=2 ELSE GOTO wert1
   mx(v)=x : my(v)=y : mz(v)=v : NEXT v
   FOR v=1 TO 40
 wert2: x=RND*18+4 : y=RND*18+4 : z=INT(RND*4+1)
   IF r(x,y,z)=0 AND p(x,y,z)=0 THEN p(x,y,z)=3 ELSE GOTO wert2
   NEXT v
   FOR v=1 TO 40
 wert3: x=RND*18+4 : y=RND*18+4 : z=INT(RND*4+1)
   IF r(x,y,z)=0 AND p(x,y,z)=0 THEN p(x,y,z)=4 ELSE GOTO wert3
   NEXT v
   FOR v=0 TO 93 : LINE(0,v)-(630,185-v),0,b : NEXT
   ON TIMER(4.29) GOSUB zeit : TIMER ON : COLOR 1,0
   xm=xb : ym=27 : r=1 : et=1 : z=0 : m=0 : ku=3
   di=1 : pu=0 : sc=0 : mi=0 : w=1 : na=185
   WINDOW OUTPUT 1 : GOSUB see
   WINDOW OUTPUT 2 : GOSUB see : GOSUB bild1 : GOTO draw
 start:  COLOR 1,0
   w=w+1 : IF w=3 THEN w=1
   LINE (565,15)-(615,15),4 : LINE (590,0)-(590,30),4
   LINE (590,15)-(590+rx(r)*25,15+ry(r)*15),7
   LOCATE 13,52 : PRINT  et  :  LOCATE 16,53 : PRINT mi":"sc
   LINE (503,1)-(556,28),4,bf : LINE (xm*2+501,ym)-(xm*2+502,ym+1),2,bf
   IF p(xm,ym,et)=2 THEN GOTO ton
   IF xm=xe AND ym=1 THEN GOSUB lift: ym=2 : GOTO draw
   LINE (506,41)-(614,74),0,bf
   LOCATE 10,51 : PRINT  pu"Punkte  ":LOCATE 11,51 : PRINT  ku"Kugeln "
   IF na<=0  THEN na=0
   IF na>185 THEN na=185
   LINE (623,0)-(631,185),0,bf : LINE (623,185)-(631,185-na),4,bf
   IF my(di)=2 THEN bild
   LOCATE 6,52 : PRINT mz(di)
   LINE (560,57)-(560+(mx(di)-xm)*2,57+(my(di)-ym)/2),7
 bild: ON w GOSUB bilda1,bilda2
   IF p(xm,ym,et)=4 THEN p(xm,ym,et)=5 : GOTO kampf
   IF p(xm,ym,et)=3 THEN p(xm,ym,et)=5 : GOTO kiste
   p(xm,ym,et)=5
 again: a=MOUSE(0) : IF a<0 THEN endmouse ELSE again
 endmouse: IF ym+ry(r)=28 THEN gameend
   IF r(xm+rx(r),ym+ry(r),et)=1 THEN xs=xm : ys=ym : q=1
   co=POINT(MOUSE(1),MOUSE(2)) : na=na-sch
   IF co=3 THEN xm=xm+rx(r) : ym=ym+ry(r) : IF q=1 THEN xm=xs:ym=ys
   IF co=2 THEN r=r-1 : IF r=0 THEN r=4
   IF co=1 THEN r=r+1 : IF r=5 THEN r=1
   IF co=5 THEN GOSUB draufsicht
 neu: IF k=5 THEN draw
      IF co=7 THEN di=di+1 : IF di>=5 THEN di=1
      IF my(di)=2 THEN k=k+1 : GOTO neu
 draw: IF na<=0 THEN m$="zu muede" : col=6 : GOSUB out : GOTO draw
      ON w GOSUB bild1,bild2
      SOUND 1000-SQR((xm-mx(di))^2+(ym-my(di))^2)*26,3,45,0
      xc=xm : yc=ym : q=0 : k=0
      LINE (0,91)-(500,187),4,bf : LINE (0,0)-(500,90),5,bf
   FOR b=1 TO 7
   IF b=1 THEN no
   xc=xc+rx(r) : yc=yc+ry(r)
 no: IF yc<1 OR yc>27 THEN loop
     n=r(xc,yc,et) : pl=p(xc,yc,et)
     IF n=1 THEN GOSUB wall : GOTO ende
     xl=xc+rxl(r) : yl=yc+ryl(r)
     IF yl<1 OR yl>27 THEN A1
     ON r(xl,yl,et)+1 GOSUB clearl,setl
 A1: xr=xc+rxr(r) : yr=yc+ryr(r)
     IF yr<1 OR yr>27 THEN A2
     ON r(xr,yr,et)+1 GOSUB clearr,setr
```

```
A2: IF b<7 THEN GOSUB lin
    IF pl<2 OR b>5 THEN nein
    ON pl-1 GOSUB ring,box,men,punkt
nein: IF xc=xe AND yc=1 THEN m$=" Lift " : col=7 : GOSUB out
loop: NEXT b
ende: GOTO start
clearl: LINE (x(2,b),y(1,b))-(x(1,b),185-y(1,b)),1,bf : RETURN
setl:   AREA (x(0,b),y(0,b))          : AREA (x(1,b),y(1,b))
        AREA (x(1,b),(185-y(1,b)))   : AREA (x(0,b),185-y(0,b))
        COLOR 2 : AREAFILL : RETURN
setr:   AREA (500-x(0,b),y(0,b))        : AREA (500-x(1,b),y(1,b))
        AREA (500-x(1,b),185-y(1,b)) :AREA (500-x(0,b),185-y(0,b))
        COLOR 2 : AREAFILL : RETURN
clearr: LINE (500-x(2,b),y(1,b))-(500-x(1,b),185-y(1,b)),1,bf : RETURN
lin:    LINE (x(1,b)-1,y(1,b))-(501-x(1,b),185-y(1,b)),4,b : RETURN
wall:   LINE (x(1,b-1),y(1,b-1))-(500-x(1,b-1),185-y(1,b-1)),3,bf
        RETURN
bild1:  WINDOW 2 :  WINDOW OUTPUT 1 : LINE (0,0)-(500,185),0,bf
        RETURN
bild2:  WINDOW 1 : WINDOW OUTPUT 2 : LINE (0,0)-(500,185),0,bf
        RETURN
bilda1: WINDOW 2 :  WINDOW OUTPUT 2 : RETURN
bilda2: WINDOW 1 :  WINDOW OUTPUT 1 : RETURN
farbe:  PALETTE 0,0,0,.5          :  PALETTE 1,.63,.63,.63
        PALETTE 2,.53,.53,.53  :  PALETTE 3,.35,.35,.35
        PALETTE 4,0,0,0           :  PALETTE 5,.27,.27,.27
        PALETTE 6,1,1,1           :  PALETTE 7,1,.87,.73
        RETURN
see:    AREA (560,145) : AREA (580,165) : AREA (540,165)
        COLOR 3 : AREAFILL
        AREA (505,175) : AREA (535,185) : AREA (535,165)
        COLOR 2 : AREAFILL
        AREA (615,175) : AREA (585,185) : AREA (585,165)
        COLOR 1 : AREAFILL
        AREA(540,175) : AREA(580,175) : AREA(580,185) : AREA(540,185)
        COLOR 5 : AREAFILL : COLOR 1
        LINE (505,40)-(615,75),4,b : LINE (505,31)-(615,39),7,bf
        LOCATE 15,55 : PRINT "Zeit" : LOCATE 13,55 : PRINT ". Stock"
        RETURN
loeschen: FOR x=1 TO 27:FOR y=1 TO 27:r(x,y,z)=1:NEXT y,x:RETURN
lift:  ON w GOSUB bilda1,bilda2
 LINE (100,20)-(250,50),1,bf : LINE (250,20)-(400,50),2,bf
 LINE (100,50)-(250,80),3,bf : LINE (250,50)-(400,80),4,bf
 COLOR 5,6 : LOCATE 3,18 : PRINT "Bitte auswaehlen"
 LOCATE 5,17 : PRINT "1" : LOCATE 5,32 : PRINT "2"
 LOCATE 8,17 : PRINT "3" : LOCATE 8,32 : PRINT "4"
 FOR x=0 TO 50 : a=MOUSE(0) : NEXT
wieder: IF MOUSE(0)=0 THEN wieder
 co=POINT(MOUSE(1),MOUSE(2)) : COLOR 4,0
 IF co<1 OR co>4 OR MOUSE(1)>500 THEN SOUND 80,1,200,0 : GOTO wieder
 FOR y=0 TO 37 : SCROLL (0,0)-(500,185),0,5
 SOUND y*50,.8,100,0 : NEXT y : et=co : RETURN
zeit: na=na+4 : sc=sc+5 : IF sc=60 THEN sc=0 : mi=mi+1
      IF mi=20 THEN gameend
      RETURN
draufsicht:     LINE(0,0)-(500,185),4,bf : COLOR 1,4
  LOCATE 2,1 : PRINT et". Stock" : d=12 : tl=0 : pu=pu-40
  FOR y=1 TO 27 : FOR x=1 TO 27
IF r(x,y,et)=0 THEN leer
v=x*d+(-6*y)-45 : c=6*y
AREA (v+200,c+6)  : AREA (v+215,c+6)  : AREA (v+215,c+15)
AREA (v+200,c+15) : COLOR 2 : AREAFILL
AREA (v+200,c+5)  : AREA (v+215,c+5)  : AREA (v+220,c)
AREA (v+205,c)    : COLOR 3 : AREAFILL
```

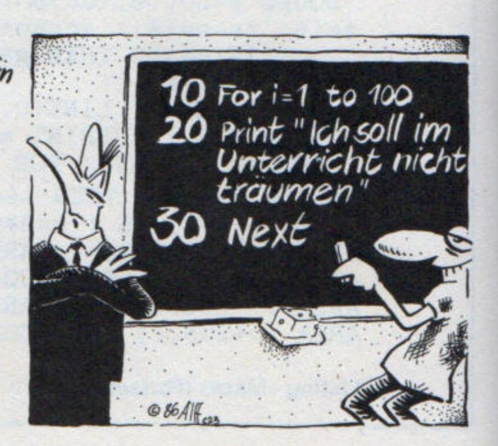
Listing »Maze« (Fortsetzung)

```
   AREA (v+215,c+15) : AREA (v+215,c+5)  : AREA (v+220,c)
   AREA (v+220,c+10) : COLOR 1 : AREAFILL
leer: NEXT x,y
   FOR v=1 TO 9 : CIRCLE(xm*d+(-6*ym)+162,6*ym-v+2),10-v,0 : NEXT
again5: IF MOUSE(0)<>0 THEN tl=tl+1
        IF tl<10 THEN again5
        RETURN
ring: CIRCLE (250,179-y(1,b-1)),7,7 : CIRCLE (250,175-y(1,b-1)),2,7
      RETURN
men:  PUT (240,171-y(1,b-1)),man,XOR : RETURN
punkt: vt=184-y(1,b-1)          : AREA (250,vt)
       AREA (250+vt/12,vt-2) : AREA (250,vt-4) : AREA (250-vt/12,vt-2)
       COLOR 2 : AREAFILL : RETURN
ton:  p(xm,ym,et)=5 : pu=pu+200
   FOR v=1 TO 4
   IF xm=mx(v) AND ym=my(v) THEN mx(v)=x1 : my(v)=2
   NEXT v
   FOR v=40 TO 500 : SOUND v,.5,100,0 : SOUND v+90,.5,100,1 : NEXT
   GOTO draw
box:  PUT (229,177-y(1,b-1)),b : RETURN
out:  COLOR col,5 : LOCATE 2,23 : PRINT m$ : RETURN
F1: m$=" Gold " : pu=pu+30 : vv=0
    FOR v=1 TO 15 : FOR col=1 TO 7 : SOUND 100*v,.2,100,INT(v/4)
    GOSUB out : NEXT col,v : GOTO draw
F2: m$=" Beamer " : col=7 : GOSUB out
    FOR v=0 TO 250 : LINE(v,0)-(500-v,187),5,b
    SOUND v*6,.2,100,0 : NEXT v
neu1: xpo=RND*18+4 : ypo=RND*18+4  : zpo=RND*3+1
    IF r(xpo,ypo,zpo)<>0 THEN neu1 ELSE xm=xpo:ym=ypo:et=zpo
    GOTO draw
F3: m$=" Zeitgewinn " : col=7 : GOSUB out : sc=0 : GOTO draw
F4: m$=" Lift " : col=7 : GOSUB out : r=1 : xm=x1 : ym=2 : GOTO draw
kiste: na=na+INT(RND*35) : ku=ku+INT(RND*2)
       ON INT(RND*9+1) GOTO F1,F1,F1,F1,F2,F3,F3,F3,F4
kampf: FOR o=0 TO 7 : FOR v=0 TO 25 : SOUND v*30,.2,o*20,0 : NEXT v,o
       IF ku<=0 THEN na=na-50 : pu=pu-100 : SOUND 100,35,255,0
       IF ku>0  THEN na=na+50 : pu=pu+100
       IF na<=0 THEN na=0
       ku=ku-1 : IF ku<=0 THEN ku=0
       GOTO draw
gameend: GOSUB bilda1 : LINE(0,0)-(631,185),5,bf
         m$="Spielende" : col=7 : GOSUB out
         IF my(1)=2 AND my(2)=2 AND my(3)=2 AND my(4)=2 THEN continue
         LOCATE 5,5 : PRINT  "Du hast nicht alle Ringe gefunden!" : END
continue: pu=pu+200+((20-mi)*100)
          LOCATE 10,5 : PRINT  "Punktestand "pu : END
```

Listing »Maze« (Schluß)







Werkzeugkasten für Superspiele

16farbige Sprites flitzen in sagenhafter Geschwindigkeit über den Bildschirm. Weitere 22 Routinen komplettieren den Programmier-Werkzeugkasten für Superspiele.

Der zweite Teil der C-Bibliothek enthält 22 Funktionen, die vor allem aus Diskettenoperationen und Farbgraphikbefehlen bestehen. Außerdem bietet die Bibliothek noch komfortable Befehle zur Abfrage des Joysticks an. Sie wurden für den C-Compiler von GST entwickelt. Er eignet sich ausgezeichnet zur Programmierung von schnellen Spielen. In Verbindung mit dem 1. Teil aus Sonderheft 6/86, Seite 114, und dem Sprite-Editor in dieser Ausgabe besitzen Sie hervorragende Programmunterstützung für schnelle Action-Spiele.

Die Routinen wurden in Assembler entwickelt und sind deshalb sehr schnell und effektiv. In Verbindung mit den 28 bereits veröffentlichten Routinen kann man auf die relativ langsamen Routinen der Standard-Bibliothek des GST-C-Compilers verzichten und dadurch wesentlich speichersparendere Programme schreiben.

Nun besitzen Sie hervorragende Instrumente, um den Spielemarkt des Atari ST zu beleben. Wir sind auch auf Ihre tollen Ergebnisse gespannt.

Dies sind alle Funktionen der 2. S&S-Bibliothek in der Übersicht:

1. start_joystick( )	;Variable als Joystickport deklarieren
2. stop_joystick( )	;Variable wieder freigeben
3. get_joystick( )	;Joystickport abfragen
4. peek( )	;Basic-Befehl Peek
5. poke( )	;Basic-Befehl Poke
6. mcs_draw( )	;16farbiges Sprite zeichnen
7. mcs4___draw( )	;4farbiges Sprite mit Farben 0 bis 3 zeichnen
8. mcs_4__draw( )	;4farbiges Sprite mit Farben 4 bis 7 zeichnen
9. mcs__4_draw( )	;4farbiges Sprite mit Farben 8 bis 11 zeichnen
10. mcs___4draw( )	;4farbiges Sprite mit Farben 12 bis 15 zeichnen
11. mcs_save( )	;Hintergrund des Sprites sichern
12. mcs_load( )	;Hintergrund des Sprites wieder laden
13. setpalette( )	;16-Farben-Palette setzen
14. setcolor( )	;Farbkomponenten einer Farbe setzen
15. gdos_create( )	;Datei erstellen
16. gdos_open( )	;Datei öffnen
17. gdos_close( )	;Datei schließen
18. gdos_read( )	;Daten lesen
19. gdos_write( )	;Daten schreiben
20. gdos_unlink( )	;Datei löschen
21. gdos_lseek( )	;Datei-Pointer setzen
22. gdos_error( )	;Fehlerbezeichnung suchen

Routinen für Profis

1. start_joystick(&port)
C-Definitionen: char port;
Die Variable port enhält nun bis auf Widerruf die Daten von Joystick-Port 1. Alle Bewegungen des Joysticks werden sofort gemeldet.
Bit 1: Oben
Bit 2: Unten
Bit 3: Links
Bit 4: Rechts

2. stop_joystick()
Beendet den automatischen Meldemodus von 1.

3. get_joystick(&port)
C-Definitionen: char port;
Die Variable port enthält nun den augenblicklichen Status des Joystickport 1.
Die Belegung gleicht start_joystick().

4. value=peek(address,len)
C-Definitionen: long address,value;
short len;
Diese Funktion liest in Value den Wert an der Adresse address mit der Länge len. Als Längen sind erlaubt:
1 = byte (.b)
2 = word (.w)
4 = longword (.l)
Mit diesem Befehl lassen sich auch privilegierte Bereiche ansprechen.

5. poke(address,value,len)
C-Definitionen: long address,value;
short len;
Diese Funktion ist das Gegenstück zu peek(). Sie schreibt den Wert value mit der Länge len an die Speicherstelle address. Es gelten dieselben Parameter wie Routine 4.
Der Wertbereich richtet sich nach der angegebenen Länge.
set_mem(&mtsx,&mem[akt_sprite*64])
C-Definitionen: char mtsx[256];
short mem[4096];

6. mcs_draw(x,y,&data)
C-Definitionen: short x,y,data[64];
Die Funktion zeichnet ein 16farbiges Sprite an die Bildschirmposition x,y. Die Daten des Sprites befinden sich im data[]-Array.
Das Format des Datenarrays:
16 Worte : plane 1
16 Worte : plane 2
16 Worte : plane 3
16 Worte : plane 4
Jeweils 4 Bit der Planes 1 bis 4 ergeben zusammen den Wert einer Farbe (0 bis 15).
Für die Files gilt folgender Aufruf:
mcs_draw(x,y,&data[18]) und
short x,y,data[18+anzahl*64];

7. mcs4__draw(x,y,&data)

C-Definitionen: short x,y,data[32];

Diese Funktion zeichnet ein 4farbiges Sprite mit den Farben 0 bis 3 (Normal-Format) an die Position x,y.

Das Daten-Format:

16 Worte : plane 1

16 Worte : plane 2

Jeweils 2 Bit der Planes 1 und 2 ergeben den Farbwert(0-3). Plane 3 und Plane 4 sind immer 0.

Die 4-Farben-Routinen laufen in Assembler zirka 40 Prozent schneller als die 16-Farben-Routine.

8. mcs__4__draw(x,y,&data)

C-Definitionen: short x,y,data[32];

wie 7., nur mit den Farben 4 bis 7.

Plane 3 ist immer 1, Plane 4 immer 0.

9. mcs__4__draw(x,y,&data)

C-Definitionen: short x,y,data[32];

wie 8., nur mit den Farben 8 bis 11.

Plane 3 ist immer 0, Plane 4 immer 1.

10. mcs__4draw(x,y,&data)

C-Definitionen: short x,y,data[32];

wie 8., nur mit den Farben 12 bis 15.

Plane 3 und Plane 4 sind immer 1.

11. mcs__save(x,y,&buffer)

C-Definitionen: short x,y,buffer[64];

Die Funktion legt den Hintergrund in der Größe eines Sprites im Array buffer[] ab.

Sie eignet sich für 4- und 16farbige Sprites.

12. mcs__load(x,y,&buffer)

C-Definitionen: short x,y,buffer[64];

Es ist das Gegenstück zur Routine 12. Der Hintergrund wird wieder auf den Bildschirm geschrieben. Das gezeichnete Sprite verschwindet.

13. setpalette(&palette)

C-Definitionen: short palette[16];

Setzt die Farbpalette des Atari ST.

Das Format:

Für jede der drei Farbkomponenten (Rot, Grün, Blau) wird hexadezimal ein Wert von 0 bis 7 eingegeben.

1. Wort : Farbe 0 (Bsp.: 0x777 = weiß)
2. Wort : Farbe 1 (Bsp.: 0x000 = schwarz)

...

16. Wort : Farbe 15 (Bsp.: 0x700 = rot)

14. setcolor(number,color)

C-Definitionen: short number,color;

Setzt die Komponenten der Farbe number.

Format gleicht der Routine 14.

15. dos=gdos__create(name,mode)

C-Definitionen: char *name;
short dos,mode;

Die Funktion kreiert eine neue Datei oder kürzt eine bestehende Datei auf Null-Länge.

name gibt den Namen der Datei an, mode die Art der Datei.

dos ist die Handle-Nummer der Datei.

mode = 0 : normale Datei
1 : »Read Only«-Datei
2 : »versteckte« Datei
4 : »versteckte« Systemdatei
8 : Datei mit »Volume-Label«

Fehlermeldungen siehe gdos__errors.

16. gdos__open(name,mode)

C-Definitionen: char *name;
short dos,mode;

Diese Funktion öffnet eine vorhandene Datei mit dem Namen name.

dos ist die Handle-Nummer der Datei.

mode = 0 : Datei ist nur lesbar.
1 : Datei ist nur beschreibbar.
2 : Datei ist lesbar und beschreibbar.

17. gdos__close(dos)

C-Definitionen: short dos;

Die Funktion schließt die Datei mit der Handle-Nummer dos.

18. err=gdos__read(address,number,dos)

C-Definitionen: * address
short err,number,dos;

Diese Funktion liest aus der Datei dos number Bytes und schreibt sie an die Adresse address.

err gibt die Anzahl der gelesenen Bytes an oder eine Fehlernummer gdos__errors()

19. err=gdos__write(address,number,dos)

C-Definitionen: *address
short err,number,dos;

Die Funktion schreibt in die Datei dos number Bytes, die im Speicher an der Position address stehen.

err gibt eine Fehlernummer zurück.

20. gdos__unlink(name)

C-Definitionen: char *name;

Diese Funktion löscht die Datei mit dem angegebenen Namen.

21. gdos__lseek(number,mode,dos)

C-Definitionen: short number,mode,dos;

Verschiebt den Dateipointer abhängig vom Modus um number Bytes.

mode ist 0 : Vom Anfang der Datei an.
1 : Von der aktuellen Position aus.
2 : Vom Ende der Datei an (number immer negativ)

22. name=gdos__error(err)

C-Definitionen: char *name;
short err;

Gibt einen String name aus, der den Fehler näher kennzeichnet.

err ist die Fehlernummer.

(Michael Schmidt/hb)

```
* SPRITES & SOUND II. LIBRARY

  Michael Schmidt
  Bottenbach 3
  7611 Berghaupten/Baden
  Tel: 07803/4864

 SECTION S.CCODE
C.1:

 xdef start_joystick
start_joystick:
 link a6,#-key.1
key.1 equ 0
 move.l 8(a6),irq_joy
 bsr getaddr
 lea 0,a0
 lea joyin,a1
 move.l a1,24(a0,d0)
 pea $14
 move.w #0,-(sp)
 move.w #25,-(sp)
```

Listing. Werkzeugkasten für Spiele

Depot-Händler

Tragen Sie Ihre Buchbestellung auf eine Postkarte ein und schicken diese an einen Depothändler in Ihrer Nähe oder an Ihren Buchhändler.

Buchhandlung Herder, Kurfürstendamm 69
1000 Berlin 15, Tel. (030) 8835002,
BTX *921782#
Computare Fachbuchhandlung, Keithstraße 18
1000 Berlin 30, Tel. (030) 2139021
Thalia Buchhaus, Große Bleichen 19
2000 Hamburg 36, Tel. (040) 3005050
Boysen + Maasch, Hermannstraße 31
2000 Hamburg 1, Tel. (040) 3005050
Electro-Data, Wilhelm-Heidsiek-Straße 1
2190 Cuxhaven, Tel. (04721) 51288
Buchhandlung Muehlau, Holtenauer Straße 116
2300 Kiel, Tel. (0431) 85085
ECL, Norderstraße 94-96
2390 Flensburg, Tel. (0461) 28181
Buchhandlung Weiland, Königstraße 79
2400 Lübeck, Tel. (0451) 160060
Buchhandlung Storm, Langenstraße 10
2800 Bremen 1, Tel. (0421) 321523
Buchhandlung Lohse-Eissing, Marktstraße 38
2940 Wilhelmshaven, Tel. (04421) 41687
Buchhandlung Schmorl u. v. Seefeld,
Bahnhofstraße 13
3000 Hannover 1, Tel. (0511) 327651
Buchhandlung Graff, Neue Straße 23
3300 Braunschweig, Tel. (0531) 49271
Deuerlich'sche Buchhandlung, Weender Straße 33
3400 Göttingen, Tel. (0551) 56868
Buchhandlung an der Hochschule,
Holländische Straße 22
3500 Kassel, Tel. (0561) 83807
Stern Verlag, Friedrichstraße 24-26
4000 Düsseldorf, Tel. (0211) 373033
Buchhandlung Baedeker, Kettwiger Straße 33-35
4300 Essen 1, Tel. (0201) 221381
Regensberg'sche Buchhandlung, Alter Steinweg 1
4400 Münster, Tel. (0251) 40541-5
Buchhandlung Acker, Johannisstraße 51
4500 Osnabrück, Tel. (0541) 28488
Buchhandlung C.L. Krüger, Westenhellweg 9
4600 Dortmund, Tel. (0231) 1527358
Buchhandlung Brockmeyer,
Querenburger Höhe 281/Unicenter
4630 Bochum, Tel. (0234) 701360
Buchhandlung Meier + Weber, Warburger Straße 98
4790 Paderborn, Tel. (05251) 63172
Buchhandlung Phönix GmbH, Oberntorwall 25
4800 Bielefeld 1, Tel. (0521) 58306-38
Buchhandlung Gonski, Neumarkt 24
5000 Köln 1, Tel. (0221) 210528
Mayer'sche Buchhandlung, Ursulinerstraße 17-19
5100 Aachen, Tel. (0241) 4777-136
Buchhandlung Behrendt, Am Hof 5a
5300 Bonn 1, Tel. (0228) 658021
Buchhandlung Cusanus, Schloßstraße 12
5400 Koblenz, Tel. (0261) 36239
Akad. Buchhandlung Interbook, Fleischstraße 61-65
5500 Trier, Tel. (0651) 43596
Buchhandlung W. Finke, Kipdorf 32
5600 Wuppertal 1, Tel. (0202) 454220
Buchhandlung Balogh, Sandstraße 1
5900 Siegen, Tel. (0271) 55298-9
Buchhandlung Naacher, Steinweg 3
6000 Frankfurt 1, Tel. (069) 298050
Buchhandlung Wellnitz, Lautenschlägerstraße 4
6100 Darmstadt, Tel. (06151) 76548
Buchhandlung Feller + Gecks, Friedrichstraße 31
6200 Wiesbaden, Tel. (06121) 304911
Ferber'sche UNI-Buchhandlung, Seltersweg 83
6300 Gießen, Tel. (0641) 12001
Sozialwissenschaftliche Fachbuchhandlung,
Friedrichstraße 24
6400 Fulda, Tel. (0661) 75077
Albertis-Hofbuchhandlung, Langstraße 47,
6450 Hanau, Tel. (06181) 24301
Gutenberg Buchhandlung, Große Bleiche 29
6500 Mainz, Tel. (06131) 37011
Buchhandlung Bock + Seip, Futterstraße 2
6600 Saarbrücken, Tel. (0681) 30677
Buchhandlung Wilhelm Hofmann,
Bismarckstraße 98
6700 Ludwigshafen, Tel. (0621) 516001
Buchhandlung Loeffler, B 1,5
6800 Mannheim 1, Tel. (0621) 28912
Buchhandlung Stehn, Bahnhofstraße 13
7000 Stuttgart 50, Tel. (0711) 561476
Osiandersche Buchhandlung, Sindelfinger Allee 25
7030 Böblingen
Buchhandlung am Markt, Kramstraße 6
7100 Heilbronn, Tel. (07131) 68682
UNI Buchhandlung Kellner + Moessner,
Kaiserstraße 18
7500 Karlsruhe, Tel. (0721) 691436
Osiandersche Buchhandlung, Wilhelmstr. 12
7400 Tübingen, Tel. (07071) 51761
Osiandersche Buchhandlung, Kaiserpassage 8
7410 Reutlingen
Buchhandlung Roth, Hauptstraße 45
7600 Offenburg, Tel. (0781) 22097
Rombach Center, Bertholdstraße 10
7800 Freiburg, Tel. (0761) 49091
Fachbuchhandlung Hofmann, Hirschstraße 4
7900 Ulm, Tel. (0731) 60949
Schauties Elektronik, Wangener Straße 99
7980 Ravensburg, Tel. (0751) 26138
Buchhandlung Hugendubel, Marienplatz
8000 München 2, Tel. (089) 2389-1
Computerbücher am Obelisk, Barerstraße 32-34
8000 München 2, Tel. (089) 282383
Pele's Computerbücher, Schillerstraße 17
8000 München 2, Tel. (089) 555229
Universitätsbuchhandlung Lachner,
Theresienstraße 43
8000 München 2, Tel. (089) 521340
Buchhandlung Schönhuber, Theresienstraße 6
8070 Ingolstadt, Tel. (0841) 33146/47
Computerstudio Gertrud Friedrich, Ludwigstraße 3
8220 Traunstein, Tel. (0861) 14767
Buchhandlung Pustet, Kl. Exerzierplatz 4
8390 Passau, Tel. (0851) 56945
Buchhandlung Pustet, Gesandtenstraße 6
8400 Regensburg, Tel. (0941)53061
Buchhandlung Dr. Büttner, Adlerstraße 10-12
8500 Nürnberg, Tel. (0911) 232318
Computer-Center-Burger, Leimitzer Straße 11-13
8670 Hof, Tel. (09281) 40075
Sortiments- u. Bahnhofsbuchh. J. Strykowski,
Bahnhofplatz 4
8700 Würzburg, Tel. (0931) 54389
Buchhandlung Pustet, Grottenau 4
8900 Augsburg, Tel. (0821) 35437
Kemptener Fachsortiment, Salzstraße 30
8960 Kempten, Tel. (0831) 14413

Schweiz:
Buchhandlung Francke AG, Neuengasse 43,
Von-Werdt-Passage
3001 Bern, Tel. (031) 221717
Buchhandlung Scherz, Marktgasse 25
3011 Bern, Tel. (031) 226837
Buchhandlung Meissner, Bahnhofstrasse 41
5000 Aarau, Tel. (064) 247151
Bücher Balmer, Neugasse 12
6300 Zug, Tel. (042) 214141
Buchhandlung Enge, Bleicherweg 56
8002 Zürich, Tel. (01) 2012078
Buchhandlung Orell Füssli, Pelikanstrasse 10
8022 Zürich, Tel. (01) 2118011
Freihofer AG, Wissenschaftliche Buchhandlung,
Universitätsstrasse 11
8033 Zürich, Tel. (01) 3634282
Buchhandlung am Rösslitor, Webergasse 5
9001 St. Gallen, Tel. (071) 228726

Österreich:
Morawa & Co, Wollzeile 11
1010 Wien, Tel. (0222) 947641
Computer Buch Shop Karl Fegerl, Heinertstraße 3
1020 Wien, Tel. (0222) 245368
Lehrmittelzentrum, Karlsplatz 13
1040 Wien, Tel. (0222) 567801
Johann Reisinger, Hauptplatz 30, Kirchenstraße 3
3302 Amstetten, Tel. (07472) 2576-0
Helmut Lainer, Obere Landstraße 8
3500 Krems, Tel. (02732) 2818
R. Pirngruber, Landstraße 34
4020 Linz, Tel. (0732) 272834
Buchhandlung Schachtner, Stadtplatz 28
4840 Vöcklabruck, Tel. (07672) 3467
R. Regelsberg, St.-Julien-Straße 2
5020 Salzburg, Tel. (0662) 73573
Tyrolia, Maria-Theresien-Straße 15
6010 Innsbruck, Tel. (05222) 24944
Wagner'sche Universitätsbuchhandlung,
Museumstraße 4
6010 Innsbruck, Tel. (05222) 22316
Buchhandlung Leykam, Stemplergasse 3
8010 Graz, Tel. (0316) 76676-0
Jos. A. Kienreich, Sacherstraße 6
8010 Graz, Tel. (0316) 76441
Volksbuchhandlung, Radetzkystraße 7
8010 Graz, Tel. (0316) 79388



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

AMIGA



Amiga
Hardware Reference Manual
Provides detailed descriptions of the graphics and audio hardware of the Amiga and explains its peripheral devices. Knowledge of assembly language is assumed.
272 Seiten/Paperback **DM 63,–**

Amiga
ROM Kernel Reference Manual: Libraries and Devices
Provides a complete listing and description of the Amiga's built-in read-only-memory (ROM) routines which support graphics, sound, and animation. Assumes a knowledge of C and assembly language.
544 Seiten/Paperback **DM 89,–**

Amiga
Intuition Reference Manual
This volume provides a complete description of Amiga's user interface, Intuition, which is used to write application programs. Assumes a knowledge of assembly language and some familiarity with the C programming language.
326 Seiten/Paperback **DM 63,–**

Amiga
ROM Kernel Reference Manual: Exec
Provides a complete listing and description of the built-in read-only-memory (ROM) routines which support Amiga's multitasking capabilities. Assumes a knowledge of C and assembly language.
176 Seiten/Paperback **DM 63,–**

alle 4 Bände zusammen

DM 260,–

THOMAS MÜLLER
COMPUTER—SERVICE
Postfach 2526
7600 Offenburg
Telefon 0781/72004

```
 trap #14
 add.l #8,sp
 unlk a6
 rts
joyin:
 move.l (irq_joy),a1
 move.b 2(a0),(a1)
 rts

 xdef irq_joy
irq_joy:
 dc.l 0

 xdef stop_joystick
stop_joystick:
 link a6,#-key.2
key.2 equ 0
 pea $15
 move.w #0,-(sp)
 move.w #25,-(sp)
 trap #14
 add.l #8,sp
 unlk a6
 rts

 xdef get_joystick
get_joystick:
 link a6,#-key.3
key.3 equ 0
 move.l 8(a6),get_joy
 bsr getaddr
 move.l d0,a0
 lea joyget,a1
 move.l a1,24(a0)
 pea $16
 move.w #0,-(sp)
 move.w #25,-(sp)
 trap #14
 add.l #8,sp
 unlk a6
 rts
joyget:
 move.l get_joy,a1
 move.b 2(a0),(a1)
 rts

 xdef get_joy
get_joy:
 dc.l 0

 xdef set_mem
set_mem:
 link a6,#-spr.1
spr.1 equ 0
 move.l 12(a6),a0
 move.l 8(a6),a1
 clr.w d0
spr.2
 clr.w d1
 clr.w d2
 clr.w d3
 clr.w d4
spr.3
 move.b (a0,d0.w),d5
 lsl.w #1,d1
 lsr.b #1,d5
 bcc spr.4
 or.w #1,d1
spr.4
 lsl.w #1,d2
 lsr.b #1,d5
 bcc spr.5
 or.w #1,d2
spr.5
 lsl.w #1,d3
 lsr.b #1,d5
 bcc spr.6
 or.w #1,d3
spr.6
 lsl.w #1,d4
 lsr.b #1,d5
 bcc spr.7
 or.w #1,d4
spr.7
 addq.w #1,d0
 move.w d0,d5
 and.w #15,d5
 bne spr.3
 move.w d1,(a1)
 move.w d2,32(a1)
 move.w d3,64(a1)
 move.w d4,96(a1)
 addq.l #2,a1
 cmp.w #256,d0
 bne spr.2
 unlk a6
 rts

 xdef getaddr
getaddr:
 move.w #34,-(sp)
 trap #14
 addq.l #2,sp
 rts

 xdef peek
peek:
 link a6,#-mem.1
mem.1 equ 0
 move.l 12(a6),a3
 clr.l -(sp)
 move.w #$20,-(sp)
 trap #1
 addq.l #6,sp
 move.l d0,savesp
 cmp.b #1,11(a6)
 bne mem.11
 move.b (a3),d7
 bra mem.14
mem.11
 cmp.b #2,11(a6)
 bne mem.12
 move.w (a3),d7
 bra mem.14
mem.12
 cmp.b #4,11(a6)
 bne mem.14
 move.l (a3),d7
mem.14
 move.l savesp,-(sp)
 move.w #$20,-(sp)
 trap #1
 add.l #6,sp
 move.l d7,a0
 clr.l d7
 unlk A6
 rts

 xdef poke
poke:
 link a6,#-mem.2
mem.2 equ 0
 move.l 16(a6),a3
 move.l 12(a6),d3
 clr.l -(sp)
 move.w #$20,-(sp)
 trap #1
 addq.l #6,sp
 move.l d0,savesp
 cmp.b #1,11(a6)
 bne mem.15
 move.b d3,(a3)
 bra mem.18
mem.15
 cmp.b #2,11(a6)
 bne mem.16
 move.w d3,(a3)
 bra mem.18
mem.16
 cmp.b #4,11(a6)
 bne mem.18
 move.l d3,(a3)
mem.18
 move.l savesp,-(sp)
 move.w #$20,-(sp)
 trap #1
 add.l #6,sp
 clr.l d7
 unlk A6
 rts

 xdef savesp
savesp:
 dc.l 0

 xdef mcs_draw
mcs_draw:
 link a6,#-mcs.1
mcs.1 equ 0
 jsr mcs_address
mcs.3
 jsr mcs_clear
 move.w (a0),d2
 move.w d2,d3
 lsr.w d0,d2
 lsl.w d1,d3
 or.w d2,(a1)
 or.w d3,8(a1)
 move.w 32(a0),d2
 move.w d2,d3
 lsr.w d0,d2
 lsl.w d1,d3
 or.w d2,2(a1)
 or.w d3,10(a1)
 move.w 64(a0),d2
 move.w d2,d3
 lsr.w d0,d2
 lsl.w d1,d3
 or.w d2,4(a1)
 or.w d3,12(a1)
 move.w 96(a0),d2
 move.w d2,d3
 lsr.w d0,d2
 lsl.w d1,d3
 or.w d2,6(a1)
 or.w d3,14(a1)

 addq.l #2,a0
 add.l #160,a1
 addq.l #1,d6
 cmp.l #16,d6
 bne mcs.3
 moveq.l #1,d6
 clr.l d7
 unlk a6
 rts

 xdef mcs_address
mcs_address:
 move.w #3,-(sp)
 trap #14
 addq.l #2,sp
 move.l d0,a1
 move.l 8(a6),a0
 move.w 18(a6),d6
 move.w 14(a6),d7
 move.w d6,d0
 and.w #15,d0
 move.w #16,d1
 sub.w d0,d1
 mulu.w #160,d7
 lsr.w #4,d6
 lsl.w #3,d6
 add.l d6,a1
 add.l d7,a1
 clr.b d6
 rts

 xdef mcs_clear
mcs_clear:
 move.w (a0),d2
 or.w 32(a0),d2
 or.w 64(a0),d2
 or.w 96(a0),d2
 move.w d2,d3
 lsr.w d0,d2
```

Listing. Werkzeugkasten für Superspiele

```
lsl.w d1,d3
eor.w #$ffff,d2
eor.w #$ffff,d3
and.w d2,(a1)
and.w d2,2(a1)
and.w d2,4(a1)
and.w d2,6(a1)
and.w d3,8(a1)
and.w d3,10(a1)
and.w d3,12(a1)
and.w d3,14(a1)
rts

xdef mcs1_clear
mcs1_clear:
move.w (a0),d2
or.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
eor.w #$ffff,d2
eor.w #$ffff,d3
and.w d2,(a1)
and.w d2,2(a1)
and.w d2,4(a1)
and.w d2,6(a1)
and.w d3,8(a1)
and.w d3,10(a1)
and.w d3,12(a1)
and.w d3,14(a1)
rts

xdef mcs2_clear
mcs2_clear:
move.w (a0),d2
or.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,4(a1)
or.w d3,12(a1)
eor.w #$ffff,d2
eor.w #$ffff,d3
and.w d2,(a1)
and.w d2,2(a1)
and.w d2,6(a1)
and.w d3,8(a1)
and.w d3,10(a1)
and.w d3,14(a1)
rts

xdef mcs3_clear
mcs3_clear:
move.w (a0),d2
or.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,6(a1)
or.w d3,14(a1)
eor.w #$ffff,d2
eor.w #$ffff,d3
and.w d2,(a1)
and.w d2,2(a1)
and.w d2,4(a1)
and.w d3,8(a1)
and.w d3,10(a1)
and.w d3,12(a1)
rts

xdef mcs4_clear
mcs4_clear:
move.w (a0),d2
or.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,4(a1)
or.w d2,6(a1)
or.w d3,12(a1)
or.w d3,14(a1)
eor.w #$ffff,d2
eor.w #$ffff,d3
and.w d2,(a1)
and.w d2,2(a1)
and.w d3,8(a1)
and.w d3,10(a1)
rts

xdef mcs_save
mcs_save:
link a6,#-mcs.4
mcs.4 equ 0
move.w #3,-(sp)
trap #14
addq.l #2,sp
move.l d0,a1
move.l 8(a6),a0
move.w 18(a6),d6
move.w 14(a6),d7
mulu.w #160,d7
lsr.w #4,d6
lsl.w #3,d6
add.l d6,a1
add.l d7,a1
clr.b d6
mcs.5
move.l 00(a1),(a0)+
move.l 04(a1),(a0)+
move.l 08(a1),(a0)+
move.l 12(a1),(a0)+

adda.l #160,a1
addq.b #1,d6
cmp.b #16,d6
bne mcs.5
move.l #1,d6
clr.l d7
unlk a6
rts
xdef mcs_load
mcs_load:
link a6,#-mcs.6
mcs.6 equ 0
move.w #3,-(sp)
trap #14
addq.l #2,sp
move.l d0,a1
move.l 8(a6),a0
move.w 18(a6),d6
move.w 14(a6),d7
mulu.w #160,d7
lsr.w #4,d6
lsl.w #3,d6
add.l d6,a1
add.l d7,a1
clr.b d6
mcs.7
move.l (a0)+,00(a1)
move.l (a0)+,04(a1)
move.l (a0)+,08(a1)
move.l (a0)+,12(a1)

adda.l #160,a1
addq.b #1,d6
cmp.b #16,d6
bne mcs.7
move.l #1,d6
clr.l d7
unlk a6
rts
xdef mcs4___draw
mcs4___draw:
link a6,#-mcs.8
mcs.8 equ 0
jsr mcs_address
mcs.9
jsr mcs1_clear
move.w (a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,(a1)
or.w d3,8(a1)
move.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,2(a1)
or.w d3,10(a1)
addq.l #2,a0
add.l #160,a1
addq.l #1,d6
cmp.l #16,d6
bne mcs.9
moveq.l #1,d6
clr.l d7
unlk a6
rts
xdef mcs_4__draw
mcs_4__draw:
link a6,#-mcs.10
mcs.10 equ 0
jsr mcs_address
mcs.11
jsr mcs2_clear
move.w (a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,(a1)
or.w d3,8(a1)
move.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,2(a1)
or.w d3,10(a1)

addq.l #2,a0
add.l #160,a1
addq.l #1,d6
cmp.l #16,d6
bne mcs.11
moveq.l #1,d6
clr.l d7
unlk a6
rts
xdef mcs__4_draw
mcs__4_draw:
link a6,#-mcs.12
mcs.12 equ 0
jsr mcs_address
mcs.13
jsr mcs3_clear
move.w (a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,(a1)
or.w d3,8(a1)
move.w 32(a0),d2
move.w d2,d3
lsr.w d0,d2
lsl.w d1,d3
or.w d2,2(a1)
or.w d3,10(a1)

addq.l #2,a0
add.l #160,a1
addq.l #1,d6
cmp.l #16,d6
bne mcs.13
moveq.l #1,d6
clr.l d7
unlk a6
rts

xdef mcs___4draw
mcs___4draw:
link a6,#-mcs.14
mcs.14 equ 0
jsr mcs_address
mcs.15
jsr mcs4_clear
move.w (a0),d2
move.w d2,d3
lsr.w d0,d2
```

```
 lsl.w d1,d3
 or.w d2,(a1)
 or.w d3,8(a1)
 move.w 32(a0),d2
 move.w d2,d3
 lsr.w d0,d2
 lsl.w d1,d3
 or.w d2,2(a1)
 or.w d3,10(a1)

 addq.l #2,a0
 add.l #160,a1
 addq.l #1,d6
 cmp.l #16,d6
 bne mcs.15
 moveq.l #1,d6
 clr.l d7
 unlk a6
 rts

 xdef setpalette
setpalette:
 link a6,#-bios.1
bios.1 equ 0
 move.l 8(a6),-(sp)
 move.w #6,-(sp)
 trap #14
 addq.l #6,sp
 unlk a6
 rts

 xdef setcolor
setcolor:
 link a6,#-bios.2
bios.2 equ 0
 move.w 10(a6),-(sp)
 move.w 14(a6),-(sp)
 move.w #7,-(sp)
 trap #14
 addq.l #6,sp
 move.l d0,a0
 unlk a6
 rts

 xdef gdos_create
gdos_create:
 link a6,#-dos.1
dos.1 equ 0
 move.w 10(a6),-(sp)
 move.l 12(a6),-(sp)
 move.w #$3c,-(sp)
 trap #1
 addq.l #8,sp
 move.l d0,a0
 unlk a6
 rts

 xdef gdos_open
gdos_open:
 link a6,#-dos.2
dos.2 equ 0
 move.w 10(a6),-(sp)
 move.l 12(a6),-(sp)
 move.w #$3d,-(sp)
 trap #1
 add.l #8,sp
 move.l d0,a0
 unlk a6
 rts

 xdef gdos_close
gdos_close:
 link a6,#-dos.3
 dos.3 equ 0
move.w 10(a6),-(sp)
move.w #$3e,-(sp)
trap #1
addq.l #4,sp
move.l d0,a0
unlk a6
rts

 xdef gdos_read
gdos_read:
link a6,#-dos.4
dos.4 equ 0
move.l 16(a6),-(sp)
move.l 12(a6),-(sp)
move.w 10(a6),-(sp)
move.w #$3f,-(sp)
trap #1
add.l #12,sp
move.l d0,a0
unlk a6
rts

xdef gdos_write
gdos_write:
link a6,#-dos.5
dos.5 equ 0
move.l 16(a6),-(sp)
move.l 12(a6),-(sp)
move.w 10(a6),-(sp)
move.w #$40,-(sp)
trap #1
add.l #12,sp
move.l d0,a0
unlk a6
rts

xdef gdos_unlink
gdos_unlink:
link a6,#-dos.6
dos.6 equ 0
move.l 8(a6),-(sp)
move.w #$41,-(sp)
trap #1
addq.l #6,sp
move.l d0,a0
unlk a6
rts

xdef gdos_lseek
gdos_lseek:
link a6,#-dos.7
dos.7 equ 0
move.w 18(a6),-(sp)
move.w 10(a6),-(sp)
move.l 12(a6),-(sp)
move.w #$42,-(sp)
trap #1
add.l #10,sp
move.l d0,a0
unlk a6
rts

xdef gdos_error
gdos_error:
link a6,#-dos.8
dos.8 equ 0
move.b 11(a6),d1
clr.b d0
lea numbers,a0
dos.9
cmp.b (a0)+,d1
beq dos.10
addq.b #1,d0
cmp.b #10,d0
bne dos.9
bra dos.11
dos.10
lea errors,a0
mulu.w #32,d0
adda.l d0,a0
bra dos.12
dos.11
lea ok,a0
dos.12
unlk a6
rts

xdef numbers
numbers:
dc.b -32,-33,-34,-35,-36,-37,-39,-40,-46,-49
xdef ok
ok:
dc.b 'no error',0

xdef errors
errors:
dc.b 'ungültige Funktionsnummer        ',0
dc.b 'Datei nicht gefunden             ',0
dc.b 'Pfadname nicht gefunden          ',0
dc.b 'zuviel offene Dateien            ',0
dc.b 'Zugriff nicht möglich            ',0
dc.b 'ungültige Handlenummer           ',0
dc.b 'nicht genügend Speicher          ',0
dc.b 'ungültige Speicherblockadresse   ',0
dc.b 'ungültige Laufwerksbezeichnung   ',0
dc.b 'keine weiteren Daten             ',0

end
```

Listing. Werkzeugkasten für Superspiele (Schluß)



Spürhund

Schon wieder ist bei der Entwicklung eines Programms der ST abgestürzt. Mit »Status« wissen Sie endlich warum.

Jeder Programmierer kann ein Lied davon singen: Da »feilt« man an einem Programm herum und ist sich sicher, daß es jetzt einwandfrei funktioniert. Aber nach dem Start zeigen sich nur die berüchtigten kleinen Bömbchen. Da hilft alles Ärgern nichts, sondern nur mit mühevoller Akribie den Fehler aufzuspüren.

Mit »Status« steht Ihnen ein ausgezeichneter Spürhund bei. Fehler erschnüffeln ist für ihn ein Kinderspiel. Die Bedienung dieses kurzen Programmes ist unkompliziert. Nach dem Absturz des Betriebssystems hilft nur ein Druck auf die Reset-Taste, um den Computer wieder arbeitswillig zu machen. Starten Sie anschließend »Status«, zeigt es Ihnen auf dem Bildschirm, welcher Fehler den Absturz produzierte, da es die Registerinhalte, die Exception-Nummer und eine Kommentarzeile ausgibt. Mit einem Tastendruck kommen Sie wieder zum Desktop zurück.

Das Programm ist vollständig in Assembler (GST) geschrieben. Es gibt den Bereich 380 (Hex) bis 3EB (Hex) auf dem Bildschirm aus. In diesem Bereich legt der ST die Werte ab, die zum Systemabsturz geführt haben.

(J. Mehnen/R. Trauner/hb)

```
         section  status
start:   move.l   sp,a5
         move.l   4(a5),a5
         move.l   $c(a5),d0
         add.l    $14(a5),d0
         add.l    $1c(a5),d0
         add.l    #$1100,d0
         move.l   d0,d1
         add.l    a5,d1
         and.l    #-2,d1
         move.l   d1,sp
         move.l   d0,-(sp)
         move.l   a5,-(sp)
         clr.w    -(sp)
         move.w   #$4a,-(sp)
         trap     #1
         add.l    #12,sp
         dc.w     $a00a
         move.l   #copy,-(sp)
         move.w   #38,-(sp)
         trap     #14
         addq.l   #6,sp
prog:    move.w   #'E',d5
         jsr      disp
         lea      text0,a3
         jsr      text
         lea      logo,a3
         jsr      text
         lea      text1,a3
         jsr      text
         lea      statin,a6
         adda.l   #4,a6
         lea      hex,a5
         move.w   #7,d4
loop3:   lea      text2,a3
         jsr      text
         move.w   #3,d6
         jsr      byteaus
         lea      text2,a3
         jsr      text
         adda.l   #$1c,a6
         tst.w    d4
         beq      jump0
         move.w   #3,d6
         jsr      byteaus
return:  lea      text2,a3
         jsr      text
         adda.l   #$24,a6
         move.w   #3,d6
         jsr      byteaus
         suba.l   #$48,a6
         move.w   #13,-(sp)
         move.w   #2,-(sp)
         trap     #1
         addq.l   #4,sp
         move.w   #10,-(sp)
         move.w   #2,-(sp)
         trap     #1
         addq.l   #4,sp
         dbra     d4,loop3
         lea      statin,a6
         lea      text4,a3
         jsr      text
         adda.l   #$40,a6
         move.w   #3,d6
         jsr      byteaus
         lea      text5,a3
         jsr      text
         adda.l   #$4,a6
         move.w   #3,d6
         jsr      byteaus
         lea      text6,a3
         jsr      text
         lea      statin,a6
         move.w   #3,d6
         jsr      byteaus
         lea      text7,a3
         jsr      text
         lea      statin,a6
         adda.l   #68,a6
         move.l   a6,-(sp)
         move.w   #0,d6
         jsr      byteaus
         move.l   (sp)+,a6
         move.b   (a6),d7
         cmpi.w   #11,d7
         bgt      zugross
         lea      excp0,a3
```

Mit »Status« spüren Sie die Ursache von Systemabstürzen auf

Wichtiger Hinweis:

Das nächste 68000'er Sonderheft erscheint am 28.11.1986

Anzeigenschluß ist der 27.10.1986

```
loop4:    cmpi.b   #0,(a3)+
          beq      schrift
          bra      loop4
schrift:  dbra     d7,loop4
          jsr      text
ende:     move.w   #7,-(sp)
          trap     #1
          addq.l   #2,sp
          clr.w    -(sp)
          trap     #1
copy:     move.w   #$6a,d0
          lea      $380,a0
          lea      statin,a1
loop1:    move.b   (a0)+,(a1)+
          dbra     d0,loop1
          rts
disp:     move.w   #27,-(sp)
          move.w   #2,-(sp)
          trap     #1
          addq.l   #4,sp
          move.w   d5,-(sp)
dispt:    move.w   #2,-(sp)
          trap     #1
          addq.l   #4,sp
          rts
byteaus:
loop2:    move.b   (a6),d7
          andi.l   #$f0,d7
          lsr.l    #4,d7
          move.b   0(a5,d7.w),d7
          move.w   d7,-(sp)
          move.w   #2,-(sp)
          trap     #1
          addq.l   #4,sp
          move.b   (a6)+,d7
          andi.l   #$0f,d7
          move.b   0(a5,d7.w),d7
          move.w   d7,-(sp)
          move.w   #2,-(sp)
          trap     #1
          addq.l   #4,sp
          dbra     d6,loop2
          rts

text:     move.l   a3,-(sp)
          move.w   #9,-(sp)
          trap     #1
          addq.l   #6,sp
          rts
jump0:    lea      text3,a3
          jsr      text
          bra      return

zugross:  lea      excph,a3
          jsr      text
          bra      ende

          section  store

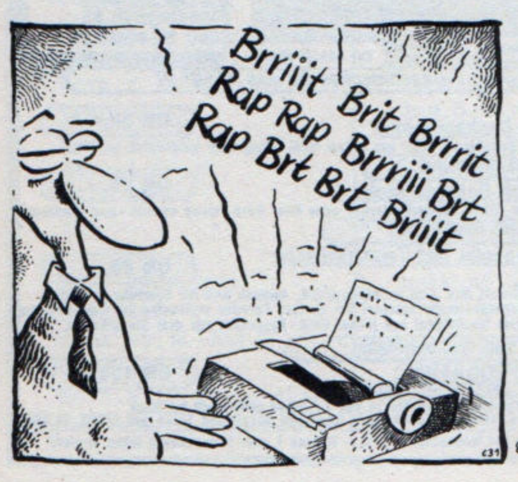
statin:   ds.b     $6b

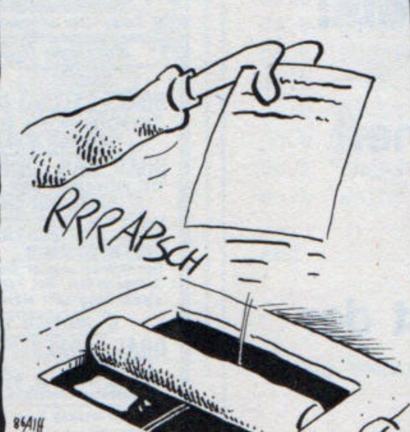
          section  data

logo:     dc.b     27,'Y',56,45,'(C) by MeTra SOFTWORKS'
          dc.b     ' für Happy-Computer und 68000er',0
hex:      dc.b     '0123456789ABCDEF'
text0:    dc.b     27,'Y',33,62,'PROCESSOR - STATUS',0
text1:    dc.b     27,'Y',35,36,'D0 - D7        A0 -
          A6          (SSP)'
          dc.b     13,10,10,0
text2:    dc.b     '    ',0
text3:    dc.b     '       ',0
text4:    dc.b     27,'Y',36,72,'SSP.....   :',0
text5:    dc.b     27,'Y',38,72,'USP.....   :',0
text6:    dc.b     27,'Y',40,72,'Nummer..   :',0
text7:    dc.b     27,'Y',42,72,'Exception :',0
excp0:    dc.b     0,27,'Y',46,72,'* Stackpointer nach
          Reset *',0
          dc.b     27,'Y',46,72,'* Programmcounter nach
          Reset *',0
          dc.b     27,'Y',46,72,'* Busfehler *',0
          dc.b     27,'Y',46,72,'* Adressfehler *',0
          dc.b     27,'Y',46,72,'* Illegaler Befehl *',0
          dc.b     27,'Y',46,72,'* Division durch
          Null *',0
          dc.b     27,'Y',46,72,'* CHK-Befehl *',0
          dc.b     27,'Y',46,72,'* TRAPV-Befehl *',0
          dc.b     27,'Y',46,72,'* Privileg-
          Verletzung *',0
          dc.b     27,'Y',46,72,'* TRACE *',0
          dc.b     27,'Y',46,72,'* Line A Emulator *',0
          dc.b     27,'Y',46,72,'* Line F Emulator *',0
excph:    dc.b     27,'Y',46,72,'* nicht definiert *',0

          end
```

Mit »Status« spüren Sie die Ursache von Systemabstürzen auf (Schluß)









Famoser Editor für flinke Sprites

Farbenprächtige Sprites zu erzeugen, ist mit diesem Editor der Superlative kein Problem. Maussteuerung und viele leistungsstarke Funktionen machen ihn zu einem Muß für jeden Spieleprogrammierer.

Der Sprite-Editor ist ein Grafik-Tool, das für die Programmierung schneller Videospiele und Grafikanimationen entwickelt wurde. Deshalb kann man das Programm um eigene Routinen erweitern. Aus diesem Grund werden bei den Beschreibungen die beeinflußten Variablen global definiert.

Bis zu 64 Sprites kann man gleichzeitig auf dem Bildschirm erzeugen. Leistungsfähige Befehle, wie »Rotate«, »Mirror« und »Animation«, gestalten das Definieren auch komplexer Sprites zu einem Kinderspiel. Mit »Animation« ruft man eine frei wählbare Anzahl Sprites in beliebiger Geschwindigkeit auf dem Bildschirm ab. Schon bei der Definition wird so geprüft, wie die Animation im fertigen Programm aussieht.

512 Farben stehen zur Wahl, um sich eine Farbpalette aus 16 Farben zu mixen. Aus diesem Grund läuft das Programm ausschließlich im niedrig auflösenden Modus. Durch verschiedene Zoom-Faktoren erscheinen auf Wunsch ein, vier oder 64 Sprites auf dem Bildschirm. Die Farbpalette und definierte Sprites kann man auf Diskette speichern. Das ermöglicht es, eine umfangreiche Bibliothek der schönsten Farbkombinationen und außergewöhnlicher Sprites zusammenzustellen und bei Bedarf abzurufen.

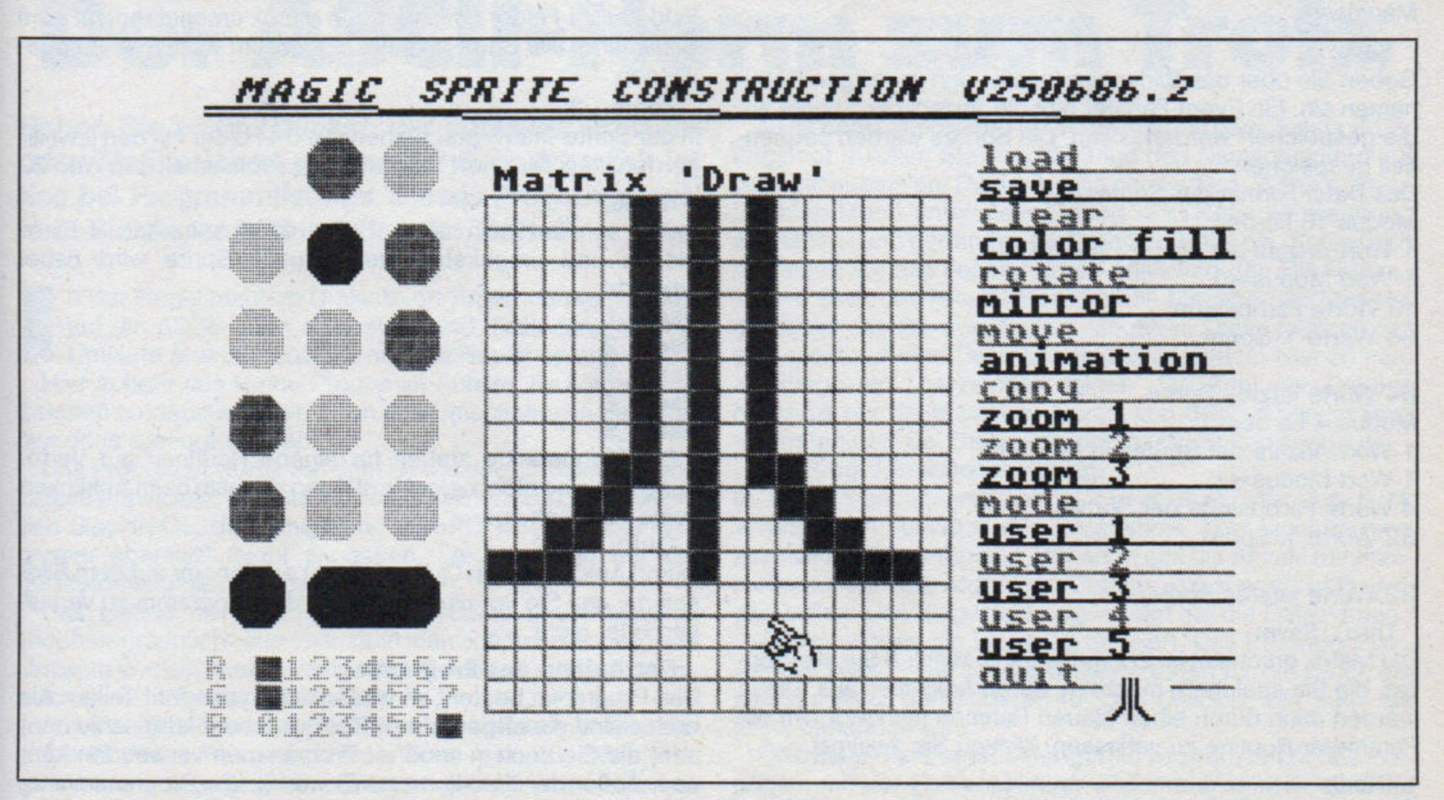
Die Maussteuerung vereinfacht die Bedienung enorm. Das Programm, mit dem C-Compiler von GST geschrieben, ergibt zirka 30 KByte Source-Code. Es würde hier unvertretbar viel Platz beanspruchen und wäre auch zum Abtippen zu umfangreich. Sie finden es deshalb mit vielen anderen interessanten Programmen auf der Leserservice-Diskette zum Atari ST. Wer das Listing dennoch haben will, kann uns schreiben. Legen Sie dann bitte einen an Sie selbst adressierten und frankierten Rückumschlag (DIN A4) bei. Adresse:
Redaktion Happy-Computer
Markt & Technik Verlag AG
»ST-Sprite-Editor«
Hans-Pinsel-Str. 2
D-8013 Haar

Programmbeschreibung:
Der Bildschirm unterteilt sich in vier Bereiche: Farb-, Einstell-, Pixel-, Befehls-Menü:

Farb-Menü:
Im Bereich 1 (links oben) können Sie die aktuelle Malfarbe aus 16 Farben auswählen.

Klicken Sie das Feld mit der gewünschten Farbe an. Die aktuelle Farbe wird dann in einem Rechteck unter den Farbfeldern angezeigt.
Die Short-Variable akt_color enthält die gewählte Farbe.

Einstell-Menü:
Hier stellen Sie die RGB-Werte der aktuellen Farbe ein. Klicken Sie für rot, grün und blau die jeweils gewünschte Intensität an. Die Farbpalette ändert sich sofort. Das Short-



ST-Sprite-Editor: Trotz Verzicht auf GEM komfortables Definieren 16farbiger Sprites



Array palette[16] enthält im GEM-Format die 16 Farbwerte. (0x000 = schwarz / 0x777 = weiß)

Pixel-Menü:
Hier reagiert das Programm je nach Zoom-Modus verschieden. Im Zoom_1-Modus können Sie den Sprite Punkt für Punkt editieren. Die einzelnen Spritepunkte werden in der aktuellen Farbe akt_color (0-15) gesetzt.

Im Zoom_2- und Zoom_3-Modus wählen Sie den aktuellen Sprite aus. Im Zoom_2-Modus können Sie unter vier Sprites wählen, im Zoom_3-Modus unter der maximalen Anzahl von 64 Sprites. Die Nummer des aktuellen Sprites läßt sich in der Short-Variablen akt_sprite (0-63) nachfragen.

Befehls-Menü:
Jeder Befehl läßt sich durch einfaches Anklicken auswählen. Einige davon benötigen Parameter, die man durch Hilfsroutinen einstellt. So muß man sie nicht bei jedem Funktionsaufruf neu eingeben. Die Hilfsroutinen löst man durch Doppelklick auf die jeweilige Funktion aus. Folgende Funktionen speichern die gewählten Parameter:
Load, Save, Animation, Copy

Routinen für jeden Zweck

Load:
Klicken Sie den gewünschten Dateinamen im Bildschirmfenster an. Ein Event-Fenster gibt die Anzahl der Sprites im ausgesuchten File an. Die Sprites werden an die gewählte Position geladen.

Dub_Load:
Nach dem Anklicken erscheint die Matrix im Zoom-Faktor 3. Wählen Sie die Position, an die der erste Sprite geladen werden soll, durch Anklicken des Feldes aus. Um den betreffenden Sprite wird ein blauer Rahmen gezeichnet. Zum Verlassen einer Parameter-Routine betätigen Sie zweimal die linke Maustaste.

Save:
Geben Sie über das Bildschirmfenster den jeweiligen Dateinamen ein. Ein Event-Fenster gibt die Anzahl der Sprites an, die gespeichert werden sollen. Die Sprites werden sequentiell gespeichert.
Das Datei-Format der Sprites:
Modus 16 Farben:
1 Wort Anzahl der Sprites in der Datei
1 Wort Modus=0
16 Worte Farbpalette
64 Worte 1. Sprite
...
64 Worte letztes Sprite
Modus 4 Farben:
1 Wort Anzahl der Sprites in der Datei
1 Wort Modus=1
4 Worte Farbpalette der Sprites
32 Worte 1. Sprite
...
32 Worte letztes Sprite

Dub_Save:
Die Matrix erscheint im Zoom-Faktor 3. Wählen Sie alle Sprites, die Sie speichern möchten, durch Anklicken aus. Diese werden dann durch einen blauen Rahmen markiert. Um die Parameter-Routine zu verlassen, klicken Sie zweimal.

Clear:
Löscht den aktuellen Sprite.

Color-Fill:
Nach dem Anklicken erscheint die Matrix im Zoom-Faktor 1. Geben Sie den Punkt ein, ab dem das Sprite gefüllt werden soll. Der Bereich wird mit der aktuellen Farbe (akt_color) »ausgemalt«.

Rotate:
Der aktuelle Sprite dreht sich im Uhrzeigersinn um 90 Grad.

Mirror:
Zwei Wahlfelder erscheinen auf der Matrix. Wählen Sie durch Anklicken zwischen horizontal und vertikal spiegeln.

Animation:
Die gewählten Sprites werden nacheinander angeschaltet. Verändern Sie die Geschwindigkeit durch Anklicken einer Position auf dem Geschwindigkeitsbalken. Je größer der gemusterte Balken, desto geringer die Geschwindigkeit. Verlassen Sie die Routine wieder mit Doppelklick.

Dub_Animation:
Wählen Sie die zu animierenden Sprites wie bei Save aus. Zum Verlassen der Parameter-Routine klicken Sie zweimal mit der linken Maustaste.

Copy:
Die Matrix erscheint im Zoom-Faktor 3. Durch den Mauszeiger und Anklicken eines Feldes bestimmen Sie, ab welcher Position die Sprites kopiert werden sollen. Die Bereiche der Sprites dürfen sich nicht überschneiden!

Dub_Copy:
Wie bei den anderen Dub-Funktionen erscheint die Matrix in Zoom-Stufe 3. Wählen Sie die zu kopierenden Sprites wie bei Save aus. Zum Verlassen der Parameter-Routine klicken Sie zweimal mit der linken Maustaste.

Zoom_1:
Die Matrix erscheint mit 16 x 16 Feldern. Befand sich in dem aktuellen Feld ein Sprite, so wird es jetzt angezeigt. In dieser Zoom-Stufe lassen sich Sprites definieren oder ändern. Die Short-Variable akt_zoom wird auf Wert 1 gesetzt.

Zoom_2:
Das Matrixfeld wird geviertelt. Vier Inhalte ab dem aktuellen Feld der 64 Felder umfassenden Matrix erscheinen auf dem Bildschirm. Die Short-Variable akt_zoom wird Wert 2 zugewiesen.

Zoom_3:
In der Sprite-Matrix erscheinen alle 64 Felder mit den jeweiligen Inhalten. Die short-Variable akt_zoom erhält den Wert 3.

Mode:
Damit schaltet man vom 16-Farb-Modus in den 4-Farb-Modus und umgekehrt. Der aktuelle Sprite wird dabei gelöscht.

User_1:
User_2:
User_3:
User_4:
User_5:
Die Menüpunkte stehen für eigene Routinen zur Verfügung. Sie sind nicht belegt und lösen deshalb beim Anklicken keine Funktion aus.

Quit:
Durch Anklicken von Quit erscheint ein Fenster auf dem Bildschirm, das Sie vor die Wahl stellt, das Programm zu verlassen oder nicht.

Der Aufbau des Programms:
Das Programm besteht im wesentlichen aus fünf Teilen. Als erstes sind die **allgemeinen Routinen** oder Utilities zu nennen, die Sie auch in anderen Programmen verwenden können. Außer der Funktion open_work(), die Programmvariablen initialisiert, und der Funktion set_pxy(), für die das glo-



bale Array pxy[4] definiert werden muß, braucht man nur lokale Variablen. Dazu kommen alle Routinen, in denen **Grafikausgaben** erfolgen; die **Eingabe-Routinen** oder auch Maus-Events, die sämtliche Mauseingaben verwalten; und die **Befehls-Routinen**, die alle Befehle enthalten, die man durch die Maus aufrufen kann. Sämtliche Manipulationen im Speicher werden durch die **Speicher-Routinen** vorgenommen, wie drehen, spiegeln, kopieren.

Neue Routinen einbinden:
Gehen Sie dabei von Teil 4, den Befehls-Routinen, aus. Von der Funktion ex__command() aus werden alle Befehle aufgerufen. Wählen Sie einen der fünf User__x()-Befehle aus, an deren Stelle Sie den Namen Ihrer Funktion eintragen. Wenn Ihre Routine eine Parameter-Routine benötigt, müssen Sie vor dem Befehlsaufruf, wie bei Save, Load, Animation und Copy, noch die Variable akt__klick abfragen. Sie zeigt an, ob der Befehl auf einfachen oder doppelten Klick reagiert. Tragen Sie den Namen Ihrer Funktion an der Stelle der User__x()-Funktion ein und eventuell noch eine Funktion mit Extention dub__ (Parameter). Die Anweisungen befinden sich vor der Funktion ex__command(). Laden Sie anschließend die Speicher-Routinen, und ändern Sie im Befehl com__namen() den Namen der User__x()-Funktion auf Ihren Funktionsnamen ab. Dieser Name erscheint dann im Bildschirmmenü.

Variablen und Funktionen:
Arrays:
Das Char-Array **»mtsx[256]«** enthält die Speicherstelle mit Farbregister (0 bis 15) für jeden Punkt des aktuellen Sprites. Punkte sprechen Sie mit der Formel »mtsx[16*y+x]« an.

Das Short-Array **»mem[4096]«** beinhaltet die Daten aller 64 Sprites. Jedes Sprite belegt 64 Wörter, die in vier Blöcke zu je 16 Wörter eingeteilt sind, wobei in jedem Block die 16 Wörter einer Bit-Plane stehen.

Char-Array **»akt__pal[3]«** speichert die RGB-Farbwerte der aktuellen Farbe.

Das Short-Array **»palette[16]«** enthält die RGB-Farbwerte im GEM-Format »akt__color« die aktuelle Farbe (0 bis 15).

In **»akt__zoom«** steht der Wert für den gültigen Zoom-Modus (1 bis 3).

Die Zahl des aktuellen Sprites (0 bis 63) findet man in **»akt__sprite«**.

Ob ein Befehl mit ein oder zwei Mausklicks angewählt wurde, wird in **»akt__klick«** abgelegt. **»akt__mode«** enthält den derzeitigen Farb-Modus. 0 entspricht 16 Farben, 1 steht für vier Farben.

In **»akt__sprcol«** wird eine Nummer abgelegt, die im Vier-Farben-Modus die Farben des Sprites anzeigt.

Funktionen:
clear__screen() (1): Löscht den gesamten Bildschirm
calcol(color) (1): Wandelt eine Farbe vom GEM-Format in Binär-Format.
colcal(color) (1): Wandelt eine Farbe vom Binär-Format ins GEM-Format.
clear__matrix() (2): Löscht die Matrix im Bildschirmmenü.
review() (2): Bringt die Matrix in Abhängigkeit vom aktuellen Zoom-Modus wieder auf den neuesten Stand.
update() (2): Gibt das aktuelle Sprite in Originalgröße am unteren, rechten Bildschirmrand aus.
set__mem(&mtsx,&mem[akt__sprite*64]) (5): Rechnet das Array »mtsx[256]« abhängig vom aktuellen Sprite »akt__sprite« in das Array »mem[4096]« um.
get__mem() (5): Ist die Umkehrfunktion zu »set__mem()«.

(Michael Schmidt/hb)

Der ASCII-Norm zum Trotz

Haben Sie einen Drucker, der entweder Umlaute oder Grafikzeichen ausdruckt? Sehr ärgerlich ist das bei Programmlistings. Dieses Programm enthebt Sie dieser Widrigkeiten der Technik.

In der Regel machen Umlaute oft Ärger, da die Drucker auf die ASCII-Norm eingestellt sind. Falls dann im Text Umlaute stehen, tauchen nur Grafikzeichen auf.

Hier schafft das kleine Programm Abhilfe. Es können alle Zeichen umdefiniert oder durch das Umschalten in den Grafikmodus erzeugt werden.

Das Programm wurde auf dem ST mit dem C-Compiler von Lattice entwickelt. An andere Computer, wie zum Beispiel an den Sinclair QL, den Amiga und IBM-PC, läßt sich das Programm ebenfalls leicht anpassen. Sie brauchen nur die geräteabhängigen Parameter ändern.

Das Listing dieses Programms druckte das Programm »höchstpersönlich« aus. Hier kann man schön erkennen, daß Umlaute und zugleich geschweifte Klammern und ein Grafikzeichen »©« verwendet wurden.

Aus Platzgründen ist die Definition auf Umlaute und ein einziges Grafikzeichen beschränkt. Dem Anwender des Programms sind dagegen keine Grenzen gesetzt, in die vollen zu gehen.

Nach dem Compilieren wird die Datei mit dem Zusatz »TTP« abgelegt. Klicken Sie anschließend den Namenseintrag auf dem Bildschirm an. Dadurch wird er invertiert. Wählen Sie im Desktop-Menü unter dem Eintrag »EXTRAS« »Anwendung anmelden« und geben Sie bei Dateityp »TOS« ein. Schließlich schließen Sie das Fenster durch Anklicken des OK-Feldes.

Dem Start des Programms folgt die Eingabe des Namens der Datei, die Sie ausdrucken möchten. Dabei müssen Sie sich nicht auf eine Datei beschränken. Jedem Namen muß allerdings ein Leerzeichen folgen. Die Zahl der Dateien begrenzt nur die maximale Anzahl von 38 Zeichen im Bildschirmfenster. Bei Dateiende erfolgt ein Seitenvorschub.

Die Programmbeschreibung:
In der Programmiersprache C bezeichnet der Ausdruck »main()« den Hauptteil des Programms. Hier tauchen zwei zusätzliche Parameter auf. Der erste gibt die Anzahl der Parameter an, also die Anzahl der Dateien, die ausgedruckt werden sollen. Das Zeigerfeld »argv« zeigt auf die Dateinamen.

Zuerst werden die Variablen definiert. Das Programm verwendet zwei Zeiger für die Dateien. Einen für die Datei, die gelesen wird, den anderen für den Drucker, der auch als Datei behandelt wird.

Falls keine Parameter eingegeben wurden, bricht das Programm mit der Fehlermeldung »Parameter fehlen« ab. Sind Dateinamen vorhanden, öffnet es den Druckerausgang als

Datei. Gelingt dies aus irgendeinem Grunde nicht, so erscheint die Fehlermeldung »Es ist keine Ausgabe auf dem Drucker möglich«.

Die folgende Schleife durchläuft alle Dateinamen und öffnet sie. Innerhalb der Schleife liest »getc« nun Zeichen für Zeichen der Eingabedatei f1. Kann ein Zeichen nicht direkt zum Drucker geschickt werden, ersetzt es eine andere Zeichenkombination. In unserem Beispiel benutzen wir den Buchstaben »ä«. Die auszugebende Kombination hängt von Ihrem Drucker ab, wir arbeiteten mit dem SP 1000 von Seikosha. Die DIP-Schalter wurden auf ASCII eingestellt. Um nun Umlaute zu erhalten, brauchen wir den deutschen Zeichensatz. In unserem Beispiel löst das beim Drucken die Kombination »27,'R',2« aus. Das Zeichen »}« wird nun auf dem Drucker durch »ä« ersetzt. Um für die nachfolgenden Zeichen wieder den ASCII-Zeichensatz zur Verfügung zu haben, wird durch die Steuerzeichen »27,'R',0« in den vorherigen Zustand umgeschaltet.

Ähnlich geht dies auch bei den folgenden Umlauten vor sich. Welche Zeichen andere ersetzen, ersehen Sie aus Ihrem Drucker-Manual.

Bei grafischen Symbolen behilft man sich, indem man den Drucker in den Grafikmodus umschaltet und die Punkte des Zeichens einzeln ausgeben läßt. Bei einem Symbol, das sieben Punkte breit ist, geschieht dies durch die Kombination »27,'K',7,0«. Die folgenden Zahlen geben an, welche Punkte gesetzt sind.

Bei grafischen Symbolen müssen Sie auf NLQ-Qualität verzichten. Sie sollten auch die Buchstabenbreite Ihres Druckers beachten, da es ansonsten zu unerwünschten Verschiebungen kommt. Der SP1000 arbeitet beispielsweise mit einer Zeichenbreite von sechs Punkten.

(Johann Gollmann/hb)

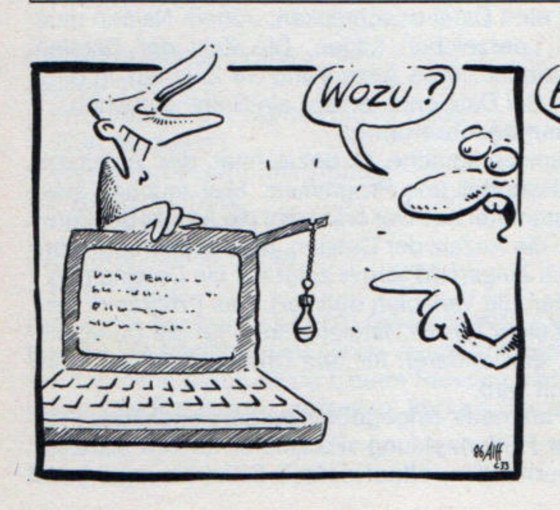
```
/***************************/
/*      Johann Gollmann    */
/*      Soft- & Hardware   */
/*      Bahnhofstraße 39   */
/*      8057 E c h i n g   */
/***************************/
#include <stdio.h>
#define page 12

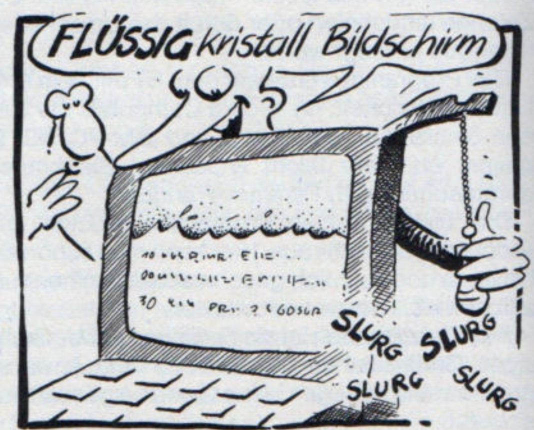
main(argc,argv)
int argc;
char *argv[];

{int i;
 char c;
 FILE *f1,*f2,*fopen();

 if (argc < 1)
 {printf("Die Parameter fehlen!\n");
  exit(1);
 }
 if ((f2 = fopen("LST:","wb")) == NULL)
 {printf("Es ist keine Ausgabe auf dem Drucker möglich\n");
  exit(1);
 }
 for (i = 1; i < argc; i++)
 {if ((f1 = fopen(argv[i],"rb")) == NULL)
  {printf("%s kann nicht geöffnet werden\n",argv[i]);
   fclose(f2);
   remove(argv[argc]);
   exit(1);
  }
  putc(page,f2);
  while ((c = getc(f1)) != EOF)
  {switch(c)
   {case 'ä':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,'{',27,'R',0);
     break;
    case 'ö':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,'| ',27,'R',0);
     break;
    case 'ü':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,'}',27,'R',0);
     break;
    case 'ß':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,'~',27,'R',0);
     break;
    case 'Ä':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,'[',27,'R',0);
     break;
    case 'Ö':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,92,27,'R',0);
     break;
    case 'Ü':
     fprintf(f2,"%c%c%c%c%c%c%c",27,'R',2,']',27,'R',0);
     break;
    case '@ ':
     fprintf(f2,"%c%c%c%c",27,'K',7,0);
     fprintf(f2,"%c%c%c%c%c%c%c",126,129,153,165,165,129,126);
     break;
    default:
     putc(c,f2);
     break;
   }
  }
  fclose(f1);
 }
}
```

Dieses kleine Programm »überlistet« die ASCII-Norm



FX-80 ST: Ein Drucker paßt sich an

Bei deutschen Umlauten oder gar beim Atari-Zeichensatz haperte es bislang an der richtigen Verständigung zwischen Drucker vom Typ Epson FX-80 und dem ST. Durch unser Listing »FX-80 ST« verstehen sich beide sofort in jeder Situation auf Pixel und Matrix.

Selbst ein Computer wie der Atari ST, der zu seinem Benutzer am liebsten in Bildern spricht, muß sich hin und wieder der Schriftsprache bedienen, um seine Mitteilungen an den Mann zu bringen. Zu diesem Zweck spendierten ihm seine Schöpfer einen derart reichhaltigen Zeichensatz, daß sich die abfällige Bezeichnung »Analphabeten-Computer« bestimmt nicht auf den ST beziehen kann. Man findet in seinem Zeichensatz neben den üblichen Buchstaben der Weltsprachen sogar Zeichen ferner Länder und Völker sowie einige sehr nützliche Grafikzeichen.

Ausgefeilte Programme wie zum Beispiel das hervorragende Textverarbeitungsprogramm »1st Word« umgehen dieses Problem mit Hilfe aufwendiger Druckertreiber, die mehr oder weniger alle Zeichen des Atari ST auf dem Drucker darstellen können. Es bietet sich jedoch eine universelle Lösung an. Die meisten Drucker besitzen nämlich einen Schreib-Lese-Speicher (RAM), in den man einen frei definierbaren Zeichensatz laden kann. Diesen »ladbaren Zeichengenerator« macht sich unser DATA-Zeilen-Listing »FX-80 ST« zunutze.

Nach dem Programmstart aus dem ST-Basic wird auf der Diskette das selbständig laufende Programm »FX-80.TOS« erzeugt. Es läßt sich wie gewohnt aus den ST-Desktop durch Doppelklick starten. »FX-80.TOS« lädt einen ST-kompatiblen Zeichensatz in den ladbaren Zeichengenerator eines FX-80- oder FX-85-Druckers. Danach kann man auch aus dem Desktop heraus oder mit Programmen wie Compilern oder Assemblern Sonderzeichen und Umlaute drucken. Unser Programm unterstützt keine NLQ-Darstellung (Near Letter Quality), wenn diese Betriebsweise einen eigenen Zeichengenerator im Drucker erfordert. Dies gilt zum Beispiel für den Fujitsu DX2100. Ferner funktioniert das Programm nur mit vollständig Epson-kompatiblen Druckern, da viele teilkompatible Drucker nur einen Teil des Zeichensatzes im RAM halten können. Beim FX-80 (und einigen Kompatiblen) muß der druckereigene RAM-Puffer vor dem Start von »FX-80 ST« abgeschaltet werden (FX-80: Dipschalter 1 bis 4 auf OFF).

Wer nun einen nicht FX-80-kompatiblen Drucker besitzt oder wer individuelle Änderungen vornehmen möchte, für den erläutern wir hier die wichtigsten Stellen im Programm. Ab Zeile 1210 sind die Codesequenzen verzeichnet, die zum Drucker gesendet werden. Die ersten beiden Byte (27,54) erweitern den ASCII-Codebereiche für druckbare Zeichen. Dies ist nötig, da sonst der Drucker die ASCII-Codes 128 bis 159 und 255 als Steuerzeichen (CR, LF, ESC etc.) interpretiert und als solche ausführt. Die fünf Byte der nächsten Zeile sorgen dafür, den Standardzeichensatz des Druckers aus dem ROM ins RAM zu kopieren. Folgende vier Zeichen (Zeile 1220) weisen den Drucker an, ab jetzt seine Zeichen aus dem Zeichensatz im RAM zu entnehmen. Die Bytes der Zeile 1221 teilen dem Drucker mit, daß nun eine Änderung der Zeichen mit dem ASCII-Code 127 bis 255 erfolgt. Ab Zeile 1230 befinden sich die Bit-Images dieser Zeichen, wobei jeweils zwölf Byte ein Zeichen definieren. Entschließt man sich dazu, weniger Zeichen umzudefinieren (oder mehr?), muß in Zeile 1070 das vierte und das fünfte Byte entsprechend geändert werden, da diese die Anzahl der zu übertragenden Bytes enthalten ($6 \times 256 + 28 = 1534$). Derartige Veränderungen erfordern unbedingt eine Anpassung der Bytes 5 und 6 in Zeile 1000 um denselben Betrag wie bei den oben genannten Bytes. Bei Änderungen der DATA-Zeilen stimmen die Prüfsummen naturgemäß nicht mehr. Deshalb muß man die Prüfsummenabfrage im Basic-Listing durch Löschen der Zeile 4040 ausschalten.

Zum Schluß noch ein Tip zum Betrieb mit »1st Word«. Da alle Sonderzeichen im Drucker definiert sind, kann bei den Hex-Dateien für die Druckertreiber-Installation die Übersetzungstabelle (Translation Table) gelöscht werden. Dies beschleunigt den Druckvorgang um einiges.

(M. Bernards/W. Fastenrath/hb)

```
30    '*           FX 80 ST             *'
40    '*                                *'
45    '*     ST-ZEICHENSATZ FUER        *'
50    '*         EPSON FX 80            *'
115   goto start
120   add:z=0
130   z=z+1
140   for i=1 to 70
150   read a:if a<0 then return
160   a(z)=a(z)+a
170   next i
180   goto 130
190   return
200   pruef:
210   for i=1 to z
220   read a
230   if a<>a(i) then goto fehler
240   next i
250   return
490   prggen:
500   open "R",#1,prgnam$,2
510   field #1, 2 as a$
520   i=0
530   i=i+1
540   read b:if b<0 then 590
550   read c:if c<0 then 590
560   d=256*b+c:lset a$=mki$(d)
570   put #1,i
580   goto 530
590   close:return
790   fehler:
800   fullw 2:clearw 2:gotoxy 0,0
810   print "FEHLER ZWISCHEN DATAZEILE";
820   print zeile + (i-1) * 100;" UND";
830   print zeile + i * 100
```

Der Atari-Zeichensatz für den Epson-Drucker

```
840   end
890   rem ***** PRUEFSUMMEN *****
900   data 2435,5516,3276,3272,2880,3598
910   data 2839,3123,3289,2168,4020,3888
920   data 3513,4722,2535,2766,2612,2526
930   data 2652,3162,2575,3319,2321,2684
940   data 1198,-1
990   rem ***** PROGRAMMDATAS *****
1000  data 096,026,000,000,006,148,000
1010  data 000,000,000,000,000,000,000
1020  data 000,000,000,000,000,000,000
1030  data 000,000,000,000,000,000,000
1040  data 063,060,000,017,078,065,084
1050  data 143,074,128,103,000,000,034
1060  data 047,060,000,000,000,119,047
1070  data 060,000,000,006,028,063,060
1080  data 000,003,063,060,000,064,078
1090  data 065,223,252,000,000,000,012
1100  data 066,103,078,065,047,060,000
1110  data 000,000,092,063,060,000,009
1120  data 078,065,062,188,000,011,078
1130  data 065,092,143,074,064,103,000
1140  data 255,186,063,060,000,007,078
1150  data 065,084,143,176,060,000,003
1160  data 102,000,255,170,066,103,078
1170  data 065,013,066,105,116,116,101
1180  data 032,068,114,117,099,107,101
1190  data 114,032,101,105,110,115,099
1200  data 104,097,108,116,101,110,000
1210  data 027,054
1211  data 027,058,000,000,000
1220  data 027,037,001,000
1221  data 027,038,000,127,255
1230  data 139,002,006,030,026
1240  data 122,098,098,026,006,002,000
1250  data 011,000,056,068,001,068,001
1260  data 070,000,068,000,000,139,000
1270  data 000,060,128,002,000,002,128
1280  data 060,002,000,139,000,028,034
1290  data 008,034,072,162,008,034,024
1300  data 000,139,000,004,010,096,010
1310  data 160,010,096,028,002,000,139
1320  data 000,004,010,160,010,032,010
1330  data 160,028,002,000,139,000,004
1340  data 010,032,138,096,010,032,028
1350  data 002,000,139,000,004,010,032
1360  data 202,096,202,096,028,002,000
1370  data 139,000,024,036,001,036,001
1380  data 038,000,036,000,000,139,000
1390  data 028,034,072,034,136,034,072
1400  data 034,024,000,139,000,028,034
1410  data 136,034,008,034,136,034,024
1420  data 000,139,000,028,034,008,162
1430  data 072,034,008,034,024,000,139
1440  data 000,000,146,000,030,000,130
1450  data 000,000,000,000,139,000,000
1460  data 082,000,158,000,066,000,000
1470  data 000,000,139,000,000,018,128
1480  data 094,000,002,000,000,000,000
1490  data 139,000,006,136,020,032,068
1500  data 032,020,136,006,000,139,000
1510  data 006,008,020,224,004,224,020
1520  data 008,006,000,139,000,062,000
1530  data 042,000,106,128,042,000,034
1540  data 000,139,000,044,002,040,002
1550  data 028,032,010,032,026,000,139
1560  data 000,062,064,144,000,254,000
1570  data 146,000,146,000,139,000,000
1580  data 028,098,000,162,000,098,028
1590  data 000,000,139,000,000,028,162
1600  data 000,034,000,162,028,000,000
1610  data 139,000,000,028,034,128,098
1620  data 000,034,028,000,000,139,000
1630  data 000,060,064,002,128,002,064
1640  data 060,002,000,139,000,000,060
1650  data 000,130,064,002,000,060,002
1660  data 000,011,000,000,057,128,005
1670  data 000,005,130,060,000,000,139
1680  data 000,000,028,162,000,034,000
1690  data 162,028,000,000,139,000,060
1700  data 130,000,002,000,002,000,130
1710  data 060,000,139,000,024,036,000
1720  data 036,195,036,000,036,000,000
1730  data 139,000,018,000,126,128,018
1740  data 128,002,128,066,000,139,000
1750  data 032,000,020,000,014,000,020
1760  data 000,032,000,139,000,062,128
1770  data 000,128,018,128,018,108,000
1780  data 000,139,000,009,000,009,054
1790  data 072,000,072,000,064,000,139
1800  data 000,004,010,032,010,096,138
1810  data 032,028,002,000,139,000,000
1820  data 018,000,094,128,002,000,000
1830  data 000,000,139,000,000,028,034
1840  data 000,098,128,034,028,000,000
1850  data 139,000,000,060,000,002,064
1860  data 130,000,060,002,000,139,000
1870  data 094,128,016,128,080,000,094
1880  data 128,000,000,139,000,094,128
1890  data 016,128,072,000,068,128,030
1900  data 000,011,000,008,020,065,020
1910  data 065,020,065,056,004,000,011
1920  data 000,056,068,001,068,001,068
1930  data 001,068,056,000,139,000,012
1940  data 002,016,002,160,002,000,002
1950  data 004,000,139,000,000,000,031
1960  data 000,016,000,016,000,016,000
1970  data 139,016,000,016,000,016,000
1980  data 031,000,000,000,000,139,000
1990  data 242,004,008,016,032,073,147
2000  data 021,009,000,139,000,227,006
2010  data 012,024,048,102,202,018,031
2020  data 000,139,000,000,000,000,223
2030  data 223,000,000,000,000,000,139
2040  data 000,016,040,068,130,016,040
2050  data 068,130,000,000,139,000,130
2060  data 068,040,016,130,068,040,016
2070  data 000,000,139,000,002,085,128
2080  data 021,128,085,008,070,129,000
2090  data 139,000,000,076,146,000,146
2100  data 064,018,076,128,000,139,001
2110  data 126,000,135,000,153,000,225
2120  data 000,126,128,139,002,022,000
2130  data 038,000,042,000,050,000,028
2140  data 032,139,000,028,034,000,034
2150  data 028,032,010,032,026,000,139
2160  data 000,124,130,000,130,254,000
2170  data 146,000,146,000,139,000,006
2180  data 008,020,128,084,000,020,008
2190  data 006,000,139,000,070,136,020
2200  data 128,084,000,084,136,006,000
2210  data 139,000,076,146,000,146,064
2220  data 018,064,146,012,000,139,000
2230  data 000,000,128,000,000,000,128
```




```
2240  data 000,000,000,139,000,000,000
2250  data 000,000,096,000,160,000,000
2260  data 000,139,000,000,032,000,124
2270  data 000,032,000,000,000,000,139
2280  data 000,096,000,240,000,148,000
2290  data 255,000,004,000,139,000,126
2300  data 000,153,000,165,036,129,000
2310  data 126,000,139,000,126,000,189
2320  data 000,169,020,129,000,126,000
2330  data 139,128,000,240,000,128,240
2340  data 000,064,000,240,000,139,000
2350  data 036,000,188,000,005,000,001
2360  data 002,188,000,139,000,132,000
2370  data 252,000,133,000,001,002,252
2380  data 000,139,000,064,004,040,048
2390  data 000,024,040,064,004,000,139
2400  data 000,068,000,068,000,068,000
2410  data 124,000,004,000,139,000,000
2420  data 000,000,004,064,008,064,016
2430  data 124,000,139,000,064,000,064
2440  data 000,064,000,124,000,064,000
2450  data 139,000,092,000,064,000,064
2460  data 000,064,000,060,000,139,000
2470  data 000,000,000,000,064,000,124
2480  data 000,000,000,139,000,000,000
2490  data 064,000,064,000,096,028,064
2500  data 000,139,000,000,000,064,060
2510  data 064,000,064,000,124,000,139
2520  data 000,124,000,004,000,004,000
2530  data 068,000,124,000,139,000,000
2540  data 000,064,000,064,000,112,000
2550  data 000,000,139,000,000,000,068
2560  data 000,068,000,068,000,056,000
2570  data 139,000,192,000,064,000,064
2580  data 000,068,008,112,000,139,000
2590  data 092,000,032,000,032,000,068
2600  data 000,060,000,139,000,000,000
2610  data 004,000,068,000,124,000,000
2620  data 000,139,000,000,000,120,000
2630  data 068,000,068,000,120,000,139
2640  data 000,004,000,124,000,004,000
2650  data 004,000,124,000,139,000,116
2660  data 000,084,000,068,000,068,000
2670  data 124,000,139,000,100,000,036
2680  data 000,020,000,012,000,100,000
2690  data 139,000,000,000,094,000,064
2700  data 000,120,000,000,000,139,000
2710  data 000,000,064,000,064,000,064
2720  data 000,060,000,139,000,124,000
2730  data 020,000,100,000,004,000,124
2740  data 000,139,000,004,000,124,000
2750  data 064,000,124,000,000,000,139
2760  data 000,000,000,000,000,064,000
2770  data 126,000,000,000,139,000,000
2780  data 000,064,000,064,000,126,000
2790  data 000,000,139,000,064,000,124
2800  data 000,068,000,068,000,124,000
2810  data 139,000,000,000,112,000,080
2820  data 000,064,000,126,000,139,000
2830  data 000,000,000,096,000,016,000,126
2840  data 000,000,000,139,000,080,170
2850  data 000,170,000,170,020,000,000
2860  data 000,139,000,006,000,012,000
2870  data 024,000,012,000,006,000,139
2880  data 028,034,000,034,020,008,020
2890  data 034,000,034,028,139,000,028
2900  data 034,000,034,020,008,020,034
2910  data 000,000,139,000,063,000,084
2920  data 000,084,000,040,000,000,000
2930  data 139,000,065,000,127,000,065
2940  data 000,096,000,112,000,139,000
2950  data 032,000,062,000,032,000,062
2960  data 000,032,000,139,000,130,000
2970  data 198,000,170,000,146,000,198
2980  data 000,139,000,028,000,034,000
2990  data 098,000,156,000,064,000,139
3000  data 000,001,000,062,000,002,000
3010  data 062,000,002,000,139,000,032
3020  data 000,062,000,032,000,032,000
3030  data 000,000,139,000,000,024,165
3040  data 000,231,000,165,024,000,000
3050  data 139,000,000,024,036,082,000
3060  data 082,036,024,000,000,139,000
3070  data 018,040,070,000,064,000,070
3080  data 040,018,000,139,000,024,004
3090  data 034,064,034,144,012,064,000
3100  data 000,139,000,024,001,036,001
3110  data 126,128,036,128,024,000,139
3120  data 000,028,000,034,000,127,000
3130  data 034,000,028,000,139,000,124
3140  data 000,146,000,146,000,146,000
3150  data 146,000,139,000,126,000,128
3160  data 000,128,000,128,000,126,000
3170  data 139,000,042,000,042,000,042
3180  data 000,042,000,042,000,139,000
3190  data 017,000,017,000,125,000,017
3200  data 000,017,000,139,000,069,000
3210  data 069,000,041,000,017,000,017
3220  data 000,139,000,017,000,017,000
3230  data 041,000,069,000,069,000,139
3240  data 000,000,000,000,000,063,064
3250  data 128,000,096,000,139,000,006
3260  data 000,001,002,252,000,000,000
3270  data 000,000,139,000,016,000,016
3280  data 068,016,068,016,000,016,000
3290  data 139,000,036,072,000,072,036
3300  data 018,000,018,036,000,139,000
3310  data 000,000,064,000,160,000,064
3320  data 000,000,000,139,000,000,000
3330  data 064,000,224,000,064,000,000
3340  data 000,139,000,000,000,000,000
3350  data 000,028,000,028,000,000,139
3360  data 008,000,004,000,002,012,048
3370  data 000,032,000,032,139,000,248
3380  data 128,000,128,000,120,000,000
3390  data 000,000,139,000,152,000,168
3400  data 000,072,000,000,000,000,000
3410  data 139,000,168,000,168,000,080
3420  data 000,000,000,000,000,139,064
3430  data 000,064,000,064,000,064,000
3440  data 064,000,064,000,000,000,000
3450  data 016,032,000,-1
4000  start:
4010  clear:restore 1000:dim a(30)
4015  for i=0 to 30:a(i)=0:next
4020  gosub add
4030  restore 900
4040  zeile=1000:gosub pruef
4050  prgnam$ = "FX80.TOS"
4060  restore 1000:gosub prggen
4070  end
4170  end
```

Der Atari-Zeichensatz für den Epson-Drucker (Schluß)

Mathematische Leckerbissen in Fortran

Studenten, Schüler und Mathematik-Freaks dürfen aufatmen. Mit unserer großen mathematischen Bibliothek in Fortran erhalten Sie eine Sammlung leistungsstarker, nützlicher Programme.

Wer professionell programmiert, der kommt ohne umfangreiche Bibliotheken nicht mehr aus. Der Programmierer steht häufig vor immer wieder gleichen Aufgaben oder vor solchen, die sich im Kern stark ähneln. In solchen Fällen wäre es natürlich verschwendete Zeit, wollte man jedesmal wieder bei Adam und Eva beginnen. Oftmals stößt man auch auf Probleme, die man selber nicht lösen kann. Es ist in unserer Zeit eben undenkbar, alles zu wissen. Deshalb greift man auf das Wissen anderer zurück. Speziell für eine Programmbibliothek bedeutet das: Man hat ein genau umrissenes Problem, beispielsweise eine partielle Differentialgleichung, jedoch fehlt entweder das ganz spezielle Fachwissen oder man weiß nicht, wie man sein Wissen in ein Programm umsetzt. Ein Unterprogramm einer Bibliothek berechnet nun die Lösung einer Differentialgleichung, ohne daß der Programmierer irgendwelche Kenntnisse über die Lösungsmethode braucht. Programmbibliotheken überbrücken aber nicht nur Wissenslücken, sondern ersparen auch Programmieraufwand und damit viel Zeit. Im Rahmen dieser Ausgabe beschränken wir uns auf die wichtigsten Programme, die in keiner mathematischen Bibliothek fehlen sollten. Die Programme sind also als Grundstock einer beliebig erweiterbaren Bibliothek zu verstehen.

Auf der Leserservice-Diskette finden Sie viele weitere nützliche Programme aus dem Bereich der Statistik. Folgende Listings haben wir zum Abtippen für Sie ausgewählt:

GAUSS (Bibliotheksprogramm Nr. 1)
Lösung eines linearen Gleichungssystems n-ter Ordnung nach dem Eliminationsverfahren von Gauss

RUKU (Bibliotheksprogramm Nr. 2)
Standard-Runge-Kutta-Verfahren zur Lösung gewöhnlicher Differentialgleichungssysteme

INTPOL (Bibliotheksprogramm Nr. 3)
Lagrange-Interpolation eines Wertes einer Funktion mit angegebenen Stützstellen

NEWTON (Bibliotheksprogramm Nr. 4)
Berechnet Nullstellen einer vorgegebenen Funktion nach dem Newton-Verfahren

SIMPSN (Bibliotheksprogramm Nr. 5)
Berechnet das Integral einer angegebenen Funktion nach der Methode von Simpson (SUBROUTINE)

ROMB (Bibliotheksprogramm Nr. 6)
Berechnet das Integral einer angegebenen Funktion nach der Methode von Romberg (FUNCTION)

MISES (Bibliotheksprogramm Nr. 7)
Bestimmung des betragsgrößten Eigenwertes und seines Eigenvektors nach Mises

MATPRO (Bibliotheksprogramm Nr. 8)
Berechnet die Produktmatrix zweier Ausgangsmatrizen

DETER (Bilbliotheksprogramm Nr. 9)
Berechnet die Determinate einer Matrix

BILD1 (Bibliotheksprogramm Nr. 10)
Zeichnet die Kurve zu einer beliebigen Funktion

FOURIER (Bibliotheksprogramm Nr. 11).
Führt die Fourieranalyse zu einer vorgegebenen Datenmenge durch

Im folgenden finden Sie zu jedem Programm eine kurze Erläuterung und jeweils ein Anwendungsbeispiel, das sich auf die Werte bezieht, die Sie jeweils im Listingteil finden. Außerdem sind die Eingangsgrößen und Ausgangsgrößen am Anfang der Listings zusätzlich aufgeführt. Die Funktion der Übergabeparameter, die am Anfang in Klammern steht, erklärt sich ebenfalls unmittelbar aus den Programmlistings.

SUBROUTINE GAUSS (MATRIX,DIM,AUS,FLAG)
Mit diesem Unterprogramm lösen Sie lineare Gleichungssysteme n-ter Ordnung. Beispielsweise sind drei Gleichungen mit den Unbekannten x, y und z gegeben:

$2x + 3y + 4z = 20$
$x + 0y + 3z = 10$
$3x + 5y - 2z = 7$

Die Matrix besteht dann aus den Koeffizienten des Gleichungssystems. Die Dimension wird gleich der Anzahl der Gleichungen gesetzt:

Matrix

2	3	4	20
1	0	3	10
3	5	-2	7

DIM = 3

Als Ergebnis erhalten Sie aus der Subroutine:

x = AUS(1) = 1
y = AUS(2) = 2
z = AUS(3) = 3
FLAG = 0

SUBROUTINE RUKU (n,xo,xe,yo,y,s)
Mit dem Programm RUKU lösen Sie Systeme gewöhnlicher Differentialgleichungen. Dem Programm muß eine Subroutine mit dem Namen FCN zur Verfügung gestellt werden, die das System der Differentialgleichungen enthält. In dem angegebenen Beispiel ist der Vektor w das Ergebnis des Differentialgleichungssystems für das Argument x und das Feld y. Lautet beispielsweise das System

(I) z1= x - y1 - 2 y2
(II) z2= x - 4 y1 + y2*y2

so lautet die zugehörige Subroutine FCN:

```
SUBROUTINE FCN(x,y,w)
REAL x,w(10),y(10)
w(1) = x - y(1) - 2 * y(2)
w(2) = x -4 * y(1) + y(2) * y(2)
RETURN
END
```

Der Aufruf des Verfahrens erfolgt mit dem Befehl CALL:

```
CALL RUKU (n,xo,xe,yo,ye,s).
```

Die Bedeutung der Parameter erklärt das Listing.

Für ein korrekt gestelltes Problem muß eine Anfangsbedingung bekannt sein. Beispielsweise ist für ein System der Lösungsvektor yo mit yo(1) = 1 und yo(2)=-1 an der Stelle xo = 0 gegeben. Gesucht ist die Lösung an der Stelle xe=4. Der Aufruf in diesem Beispiel lautet:

```
CALL RUKU (2,0,4,yo,ye,100).
```

Die 100 ist der Variablenwert für die Genauigkeit der Berechnung. Das oben genannte Beispiel wird mit dem Programm ».MAIN__RUKU.« ausgeführt. (Auch die Subroutine FCN ist dazu notwendig!) Das Testbeispiel liefert das Ergebnis:
ye(1) = 16.299736
ye(2) = 8.251969

SUBROUTINE INTPOL (X,Y,DIM,ER)
Dieses Unterprogramm berechnet den Funktionswert einer nicht bekannten Funktion an der Stelle ER. Dafür muß eine bestimmte Anzahl von x/y-Paaren des Funktionsverlaufes bekannt sein. Die Anzahl der Paare wird mit DIM an das Programm übergeben und mit den eindimensionalen Feldern X(DIM) und Y(DIM) die Funktionspaare. Der interpolierte Funktionswert wird mit der Variablen ER zurückgegeben.

Im Testbeispiel im Listing sind vier Wertepaare vorgegeben. Der Funktionswert an der Stelle ER = -0.5625 ist gesucht. Wie man leicht an den Beispielwerten erkennt, stammen die x/y-Werte aus einer Sinuskurve. Das exakte Ergebnis ist ER = -0.00982. INTPOL liefert den Näherungswert ER = -0.0118. Die Abweichung beträgt nur 0.00198. Man sieht, daß schon bei nur vier Stützstellen das Ergebnis sehr genau ist.

SUBROUTINE NEWTON
(EIN,EPS,MAXITE,ITZAHL,K,ERGEB)
Diesem Programm muß eine FUNCTION zur Verfügung gestellt werden, in der die Funktion definiert ist. Sie hat den Namen F(y):

```
FUNCTION F(Y)
F= Y*Y-4
RETURN
END
```

In diesem Beispiel handelt es sich um eine Normalparabel mit dem Scheitelpunkt bei (0/-4). Die exakten Nullstellen sind x1=-2 und x2=2. Dem Unterprogramm muß ein Wert in der Nähe der Nullstelle vorgegeben werden. Dieser Wert ist im Beispiel EIN = 1,0. Weiterhin kann die maximale Zahl der Iterationen festgelegt werden. Das heißt, wie oft das Unterprogramm die Berechnungen durchführen soll, um eine Näherung zu erhalten.

Liegt keine Konvergenz vor, so endet das Programm in einer Endlosschleife. Dieses wird aber durch die Maximalzahl der Iterationen MAXITE verhindert. In diesem Fall wird die Näherung der Nullstelle ERGEB zurückgegeben, sowie die Anzahl der Iterationen ITZAHL, um das Ergebnis bis auf eine Abweichung EPS genau zu bestimmen. Das Programm ».MAIN__NEWTON.« führt das oben genannte Beispiel aus.

FUNCTION ROMB (START,ENDE,EPS)
Die Funktion ROMB integriert eine Funktion über einen definierten Bereich. Dazu muß die Funktion in einer FUNCTION F(A,J,H) angegeben werden:

```
FUNCTION F(A,J,H)
X = A + J * H
F= X * X + 4
RETURN
END
```

In der dritten Programmzeile ist die Funktion einzutragen. Es handelt sich bei diesem Beispiel wieder um eine verschobene Normalparabel. Diese soll in den Grenzen von A = 0 bis B = 1 integriert werden. Das exakte Ergebnis läßt sich analytisch berechnen und lautet: I = 4.3333

Die Funktion ROMB wird mit der linken (START) und rechten (ENDE) Grenze aufgerufen, sowie einem EPS, das die Genauigkeit der Berechnungen festlegt. Je kleiner das EPS, desto genauer ist die Berechnung. Das Programm .MAIN__ROMBERG..« führt das Beispiel aus.

SUBROUTINE SIMPSN (LINGRE,REGRE,EPS,INTSUM)
Die Subroutine SIMPSN integriert ebenso wie ROMB. Es handelt sich hierbei jedoch um eine Subroutine. Außerdem ist die Lösungsmethode eine andere. In der FUNCTION FX wird die Funktion definiert:

```
FUNCTION FX(X)
FX = X * X + 4
RETURN
END
```

Ebenso wie bei ROMB müssen in gleicher Reihenfolge die linke Grenze LINGRE, die rechte Grenze REGRE und die Kontrollzahl für die Genauigkeit der Berechnung übergeben werden. Das Ergebnis der numerischen Integration wird mit der Variablen INTSUM an das aufrufende Programm zurückgegeben.

SUBROUTINE MISES
(MATRIX,DIM,MAXIT,EIWB,REST,EIVEK)
Dieses Unterprogramm sucht nach dem Verfahren von Mises den größten Eigenwert, sowie den zugehörigen Eigenvektor einer symmetrischen Matrix. Dem Unterprogramm muß dazu die Matrix (MATRIX), ihre Dimension (DIM) sowie die Anzahl der gewünschten Iterationen (MAXIT) übergeben werden. Die Subroutine MISES übergibt dann dem aufrufenden Programm den größten Eigenwert EIWB, den zugehörigen Eigenvektor EIVEK sowie die maximale Abweichung vom genauen Wert. Für die im Beispiel übergebene Matrix

$$A = \begin{matrix} 3.0 & 3.0 & 1.0 \\ 3.0 & 5.0 & 0.2 \\ 1.0 & 0.2 & 4.0 \end{matrix}$$

ergibt sich für den größten Eigenwert EIWB = 7.308474, für die Abweichung REST = 2.0792E-02. Der zugehörige Eigenvektor lautet:

$$EIVEK = \begin{matrix} 2.627 \\ 3.462 \\ 1.000 \end{matrix}$$

SUBROUTINE MATPRO
(MATRXA,DIMA,MATRXB, DIMB,MATRXC,DIMC,SPLT,ZE)
Dieses Unterprogramm berechnet das Produkt zweier Matrizen. Dazu ist es vorher notwendig, die Ausgangsmatrizen in ein eindimensionales Feld zu speichern. Aus einer 2x3-Matrix entsteht dann beispielsweise ein Feld mit sechs Elementen (REAL FELD(6)). Diese Felder werden dem Unterprogramm übergeben. Außerdem muß die Form der Ergebnismatrix in ZE und SPLT festgelegt werden. Multipliziert man beispielsweise eine 3x4-Matrix mit einer 4x7-Matrix, so hat die Ergebnismatrix die Größe 3x7. Es muß also die Spaltenzahl SPLT = 3 und die Zeilenzahl ZE = 7 der Subroutine MATPRO als Parameter übergeben werden.

In dem Beispiel im Listingteil ».MAIN__MATPRO.« haben die Ausgangsmatrizen eine 2x2-Form, das heißt, daß auch die Ergebnismatrix diese Form hat:

$$A = \begin{matrix} 2.0 & 0.0 \\ 0.0 & 2.0 \end{matrix} \quad B = \begin{matrix} 15.0 & 12.0 \\ 1.0 & 5.0 \end{matrix}$$

=> MATRXA(4), MATRXB(4), MATRXC(4)

$$C = \begin{matrix} 30.0 & 24.0 \\ 2.0 & 10.0 \end{matrix}$$

FUNCTION DETER (MATRIX,DIMQUA,DIM)
Diese Funktion berechnet die Determinante einer quadratischen Matrix. Dazu bringt der Algorithmus die Matrix auf eine Dreiecksgestalt nach dem Eliminationsverfahren von Gauss und ordnet die Elemente nach einer Pivotsuche. Aus der Hauptdiagonalen wird dann die Determinante berechnet. Der



Anwender hat nur die Ausgangsmatrix spaltenweise in ein eindimensionales Feld zu speichern. Der Funktion muß erst dieses Feld (MATRIX), dann die Länge des Feldes (DIMQUA) und zum Schluß die Dimension der Matrix (DIM) übergeben werden. Der beim Aufruf übergebene Wert ist dann die Determinante der Matrix.

SUBROUTINE BILD1

(XMIN,XMAX,YMIN, YMAX,FLAG,N,XE,YE,PUNKT,AN)

Das Unterprogramm BILD1 stellt zweidimensionale Funktionen oder auch einzelne Datenpunkte dar. Der Aufruf erfolgt über den Befehl:

CALL BILD1
(XMIN,XMAX,YMIN,YMAX, FLAG,N,XE,YE,PUNKT,AN)

Die Parameter haben die folgenden Bedeutungen.

XMIN: Untere Bereichsgrenze der X-Werte

XMAX: Obere Bereichsgrenze der X-Werte

YMIN: Untere Grenze der Funktionswerte

YMAX: Obere Grenze der Funktionswerte

FLAG: Ist das Flag = 1, so wird eine automatische Bereichswahl durchgeführt, das heißt die Werte YMIN und YMAX werden optimiert.

N: Anzahl der Stützstellen, durch die die Funktion dargestellt wird. Im allgemeinen reicht hier ein Wert zwischen 640 und 1000 aus, da die horizontale Bildschirmauflösung ebenfalls »nur« 640 Punkte beträgt. Zu große Werte erhöhen die Rechenzeit unnötig, während zu kleine Werte den Graphen eckig erscheinen lassen. Das Maximum von N ist 5000.

XE: Abstand der Einheiten auf der X-Achse. Die X-Achse wird von XMIN mit Einheiten im Abstand XE versehen.

YE: Abstand der Einheiten auf der Y-Achse. Die Y-Achse wird ab YMIN mit Einheiten im Abstand YE versehen.

PUNKT: Hier sind maximal 99 darzustellende Punkte gespeichert.
PUNKT(...,1): X-Koordinate PUNKT(...,2): Y-Koordinate

AN: In dieser Variable steht die Anzahl der darzustellenden Punkte.

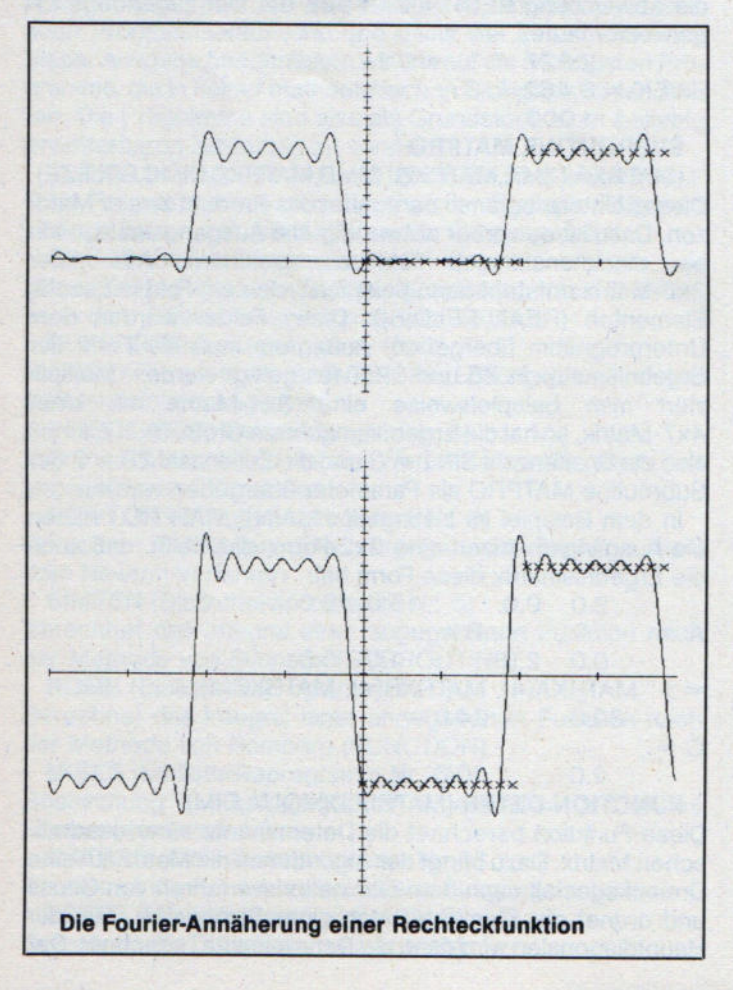
Außerdem muß die darzustellende Funktion in FCN(X) stehen:

```
FUNCTION FCN(X)
FCN = SIN(X)
RETURN
END
```

Wie Sie im Listing sehen, sind die VDI-Aufrufe ähnlich denen in C. Allerdings dürfen Sie das obligatorische CALL vor den Unterprogrammaufrufen nicht vergessen. Außerdem müssen die übergebenen Variablen vom Typ INTEGER*2 sein, da dieser Typ das Äquivalent zum SHORT-Typ in C darstellt. Auch ist zu beachten, daß die untere Grenze von Arrays die 0 ist und nicht, wie in Fortran üblich, die 1. Da bei den Unterprogrammaufrufen keine Variablen zurückgegeben werden, muß die Übergabe des HANDLE durch die Funktion AESRET() erfolgen.

Das Programm **FOURIER** stellt ein Programm zur Fourier-analyse einer gegebenen Datenmenge dar. Es berechnet aus den eingelesenen Daten eine Fourier-Reihe, die sich dann zusammen mit den Datenpunkten durch das Bibliotheksprogramm BILD1 grafisch darstellen läßt. Die Abbildung zeigt die Fourier-Annäherung einer Rechteckfunktion.

(Christian Träger/Michael Zwenger/Matthias Rosin/hb)



Die Fourier-Annäherung einer Rechteckfunktion

```
Eingangs-, Ausgangsgrößen:

   DIM      ( INTEGER                 )   : Anzahl der Gleichungen
   MATRIX   ( REAL MATRIX(DIM,DIM+1) )   : Matrix mit den Koeffizienten
                                            und dem Ergebnis
   AUS      ( REAL AUS(DIM)           )   : Ergebnisvektor
   FLAG     ( INTEGER                 )   : FLAG = -1, wenn es keine
                                            Lösung gibt, sonst FLAG = 0

C Bibliotheksprogramm Nr. 1
C*************************************************************
         SUBROUTINE GAUSS (MATRIX,DIM,AUS,FLAG)
C*************************************************************
C  - Michael Zwenger -   2913 Apen    Am Kirchweg 22    Januar 1986
C
       INTEGER DIM,FLAG,I,J,IM1,M,MM,NP1,K,L
       REAL MATRIX(DIM,DIM+1),AUS(DIM),R,TAUSCH
       FLAG = 0
       NP1 = DIM + 1
       DO 20 I=2,DIM
       DO 20 J=I,DIM
       IF (MATRIX(I-1,I-1).NE.0) GOTO 60
       IM1 = I - 1
       DO 21 M=I,DIM
       IF (MATRIX(M,IM1).EQ.0) GOTO 21
       DO 22 MM=IM1,NP1
       TAUSCH = MATRIX(M,MM)
       MATRIX(M,MM) = MATRIX(IM1,MM)
 22    MATRIX(IM1,MM) = TAUSCH
 21    CONTINUE
       FLAG=-1
       GOTO 50
 60    R = MATRIX(J,I-1) / MATRIX(I-1,I-1)
       DO 20 K=I,NP1
 20    MATRIX(J,K) = MATRIX(J,K) - R * MATRIX(I-1,K)
       DO 30 I=2,DIM
       K= DIM - I + 2
       R = MATRIX(K,NP1) / MATRIX(K,K)
       DO 30 J=I,DIM
       L = DIM - J + 1
 30    MATRIX(L,NP1) = MATRIX(L,NP1) - R * MATRIX(L,K)
       DO 90 I=1,DIM
       AUS(I) = MATRIX(I,NP1) / MATRIX(I,I)
 50    RETURN
       END
```

Listing 1. Gauss-Elimination

```
Eingangs-, Ausgangsgrößen:

         n      ( INTEGER    )    : Anzahl der Gleichungen
         xo     ( REAL       )    : Startargument
         yo(n)  ( REAL yo(n) )    : Startvektor
         xe     ( INTEGER    )    : Endwert
         s      ( INTEGER    )    : Anzahl der Schritte
         ye(n)  ( REAL ye(n) )    : Ergebnisvektor

C-----------------------------------.MAIN_RUKU.---------------------
      Integer n,i
      Real xo,xe,yo(10),ye(10)
      xo=0.
      yo(1)=1.
      yo(2)=-1.
      n=2
      xe=4.0
      Call RUKU(n,xo,xe,yo,ye,100)
      Write(*,11) 'Ergebnisvektor an der Stelle xe = ',xe,' :'
      Print*
      Do 10 i=1,n
        Write(*,12) 'y(',i,') = ',ye(i)
   10 Continue
   11 Format(3x,A35,F4.2,A2)
   12 Format(5x,A2,I2,A4,F12.6)
      End

      Subroutine FCN(x,y,w)
      Real x,w(10),y(10)
      w(1)=x-y(1)+2*y(2)
      w(2)=x+4*y(1)-y(2)*y(2)
      Return
      End

C Bibliotheksprogramm Nr. 2
C******************************************************************
                 Subroutine RUKU (n,xo,xe,yo,y,s)
C******************************************************************
C  - Michael Zwenger -     2913 Apen    Am Kirchweg 22     Mai 1986
C
      Integer n,s,i,j
      Real    xo,xe,h,x,yo(10),y(10),k1(10),k2(10),k3(10),k4(10)
        h=(xe-xo)/s
        x=xo
        Do 19 i=1,n
          y(i)=yo(i)
   19   Continue
      Do 25 i=1,s
        Call FCN(x,y,k1)
        Do 20 j=1,n
          k1(j)=h*k1(j)/2+y(j)
   20   Continue
        Call FCN (x+h/2,k1,k2)
        Do 21 j=1,n
          k2(j)=h*k2(j)/2+y(j)
   21   Continue
        Call FCN (x+h/2,k2,k3)
        Do 22 j=1,n
          k3(j)=h*k3(j)+y(j)
   22   Continue
        Call FCN (x+h,k3,k4)
        Do 23 j=1,n
          k4(j)=h*k4(j)
   23   Continue
        x=i*h+xo
        Do 24 j=1,n
          y(j)=-y(j)/3+(k1(j)+2*k2(j)+k3(j)+k4(j)/2)/3
   24   Continue
   25 Continue
      Return
      End
```

Listing 2. Das Runge-Kutta-Verfahren

```
Eingangs-, Ausgangsgrößen:

    DIM  ( INTEGER     )   : Anzahl der Funktionspaare ( >1 )
    X    ( REAL X(DIM) )   : X-Werte
    Y    ( REAL Y(DIM) )   : Y-Werte
    ER   ( REAL        )   : Übergabe der Stelle an INTPOL, an der
                             zu interpolieren ist, Rückgabe des
                             Funktionswertes an .MAIN.

C--------------------------------.MAIN_INTPOL.----------------------
      REAL A(4),B(4),ER
      INTEGER M
      M=4
      A(1)=-0.900
      A(2)=-0.675
      A(3)=-0.450
      A(4)=-0.225
      B(1)=-1.0000
      B(2)=-0.9239
      B(3)=-0.7071
      B(4)=-0.3827
      ER=  -0.5625
      CALL INTPOL(A,B,M,ER)
      PRINT*,' ',ER,' = ER'
      END

C Bibliotheksprogramm Nr. 3
C******************************************************************
             SUBROUTINE INTPOL (X,Y,DIM,ER)
C******************************************************************
C  - Michael Zwenger -   2913 Apen   Am Kirchweg 22    Januar 1986
C
      INTEGER DIM,I,J
      REAL X(DIM),Y(DIM),SUM,HILFE
      SUM=0.0
      DO 20 I=1,DIM
      HILFE=Y(I)
      DO 10 J=1,DIM
      IF ((I-J).EQ.0) GOTO 10
      HILFE=HILFE*(ER-X(J))/(X(I)-X(J))
   10 CONTINUE
   20 SUM=SUM+HILFE
      ER=SUM
      RETURN
      END
```

Listing 3. Interpolation nach Lagrange

```
Eingangs-, Ausgangsgrößen:

    EIN     ( REAL     )    : Startwert
    EPS     ( REAL     )    : zugelassene Abweichung vom exakten Wert
    MAXITE  ( INTEGER  )    : maximale Iterationszahl
    ITZAHL  ( INTEGER  )    : benötigte Iterationsschritte
    K       ( INTEGER  )    : Kontrollflag für Abbruch der Iteration
    ERGEB   ( REAL     )    : gefundene Näherungslösung

C ---------------------------------------.MAIN_NEWTON.---------------
      REAL X0,EPS,E
      INTEGER MAXIT,L,K
      X0=1.00
      EPS=0.0005
      MAXIT=50
      CALL NEWTON(X0,EPS,MAXIT,L,K,E)
      IF ((K-1).NE.0) GOTO 20
      PRINT*,'KEINE LÖSUNG'
      GOTO 30
   20 WRITE(*,10) E,L
   10 FORMAT(3X,'LÖSUNG X=',F7.2,' NACH ',I3,' ITERATIONEN')
   30 END

C Bibliotheksprogramm Nr. 4
C******************************************************************
           SUBROUTINE NEWTON (EIN,EPS,MAXITE,ITZAHL,K,ERGEB)
C******************************************************************
C  - Michael Zwenger -   2913 Apen    Am Kirchweg 22     Januar 1986
C
      REAL EPS,ERGEB,EIN,H,Y1,Y2
      INTEGER MAXITE,ITZAHL,K
      ITZAHL=0
      K=0
    1 Y1=F(EIN)
      Y2=(F(EIN+0.001)-F(EIN))/0.001
      H=Y1/Y2
      IF ((ABS(H)-EPS).GE.0) GOTO 3
    2 ERGEB=EIN
      RETURN
    3 EIN=EIN-H
      ITZAHL=ITZAHL+1
      IF ((ITZAHL-MAXITE).LE.0) GOTO 1
      K=1
      GOTO 2
      END
```

Listing 4. Nullstellensuche nach Newton

```
Eingangs-, Ausgangsgrößen:

    START   ( REAL )          : linke Intervallgrenze
    ENDE    ( REAL )          : rechte Intervallgrenze
    EPS     ( REAL )          : Kontrollzahl für die Genauigkeit
                                der Berechnung

C---------------------------------.MAIN_ROMBERG.--------------------
```

Listing 5. Integration nach Romberg

```
      REAL I
      I=ROMB(0.0,1.0,0.0001)
      PRINT*,I
      END

C Bibliotheksprogramm Nr. 5
C*****************************************************************
              FUNCTION ROMB(START,ENDE,EPS)
C*****************************************************************
C  - Michael Zwenger -   2913 Apen    Am Kirchweg 22    Januar 1986
---3----
C
      REAL STEP,ENDE,START,FELD(120),S
      INTEGER NR,ZWSCH1,ZWSCH,ZWSCH2,J,J1,L,K,M
      K = 0
      NR = 32
      STEP = ENDE - START
      G = F(START,0,STEP)
      FELD(1) = (G + F(START,1,STEP) * STEP) / 2.0
    4 K = K + 1
      STEP = 0.5 * STEP
      S = 0.0
      M = 2 ** (K-1)
      DO 1 J=1,M
      J1 = J + J - 1
    1 S = S + F(START,J1,STEP)
      FELD(K+1) = FELD(K) / 2 + STEP * S
      DO 2 L=1,K
      ZWSCH = K + 1 - L
      ZWSCH1 = ZWSCH + ( L * ( NR - L - 1 )) / 2
      ZWSCH2 = ZWSCH + (( L - 1 ) * ( NR - L )) / 2
      G = 4 ** L
    2 FELD(ZWSCH1) = ( G * FELD( ZWSCH2 + 1 )
      - FELD(ZWSCH2) )/( G - 1.0 )
      IF( (ABS(FELD(ZWSCH1) - FELD(ZWSCH2)) - EPS).GT.0) GOTO 4
    3 L = ZWSCH1 - ZWSCH + 1
      ROMB = FELD(L)
      RETURN
      END
```

Listing 5. Integration nach Romberg (Schluß)

```
Eingangs-, Ausgangsgrößen:

      LINGRE    ( REAL )    : linke Intervallgrenze
      REGRE     ( REAL )    : rechte Intervallgrenze
      EPS       ( REAL )    : Kontrollzahl für die Genauigkeit
      INTSUM    ( REAL )    : Ergebnis der Integration

C------------------------------------.MAIN_SIMPSON.-----------------
      REAL S,A,B,EPS
      A=0.0
      B=1.0
      EPS=0.00001
      CALL SIMPSN (A,B,EPS,S)
      PRINT*,S
      END

C Bibliotheksprogramm Nr. 6
C*****************************************************************
              SUBROUTINE SIMPSN (LINGRE,REGRE,EPS,INTSUM)
C*****************************************************************
C  - Michael Zwenger -   2913 Apen    Am Kirchweg 22    Januar 1986
C
      REAL LINGRE,REGRE,EPS,INTSUM,DX,X1,X2,ZWEIDX,EXFL
      INTEGER I,N,NM2
      EXFL = 0.0
      NR = 4
    1 DX = (REGRE-LINGRE) / (0.0 + NR)
      ZWEIDX = 2.0 * DX
      X1 = LINGRE - DX
      X2 = LINGRE
      INTSUM = FX(LINGRE) + FX(REGRE)
      NM2 = NR - 2
      DO 2 I=2,NM2,2
      X1 = X1 + ZWEIDX
      X2 = X2 + ZWEIDX
    2 INTSUM = INTSUM + 4.0 * FX(X1) + 2.0 * FX(X2)
      INTSUM = INTSUM + 4.0 * FX(REGRE - DX)
      INTSUM = INTSUM * DX / 3.0
      IF(ABS(INTSUM-EXFL).LE.EPS) RETURN
      EXFL = INTSUM
      NR = 2 * NR
      IF(NR.LE.2000) GOTO 1
      RETURN
      END
```

Listing 6. Integration nach Simpson

```
Eingangs-, Ausgangsgrößen:

      DIM      ( INTEGER              )  : Dimension der Matrix
      MATRIX   ( REAL MATRIX(DIM,DIM) )  : symmetrische Matrix
      MAXIT    ( INTEGER              )  : Anzahl der Iterationen
      EIVEK    ( REAL EIVEK(DIM)      )  : Eigenvektor zu EIWB
      EIWB     ( REAL                 )  : größter Eigenwert
      REST     ( REAL                 )  : Abweichung

C-------------------------------------.MAIN_MISES.--------------------
      REAL A(3,3),EWERB,REST,X(3)
      INTEGER MAXIT,DIM,I
      A(1,1)=3.0
      A(1,2)=3.0
      A(1,3)=1.0
      A(2,1)=3.0
      A(2,2)=5.0
      A(2,3)=0.2
      A(3,1)=1.0
      A(3,2)=0.2
      A(3,3)=4.0
      MAXIT=10
      DIM=3
      CALL MISES(A,DIM,MAXIT,EWERB,REST,X)
      PRINT*,'GROESSTER EIGENWERT = ',EWERB
      PRINT*,'REST = ',REST
      WRITE(*,20) (X(I),I=1,DIM)
   20 FORMAT(F10.3)
      END

C Bibliotheksprogramm Nr. 7
C*******************************************************************
      SUBROUTINE MISES (MATRIX,DIM,MAXIT,EIWB,REST,EIVEK)
C*******************************************************************
C  - Michael Zwenger -      2913 Apen    Am Kirchweg 22    Mai 1986
C
      REAL MATRIX(DIM,DIM),EIVEK(DIM),EIWA,EIWB,REST,HFELD(20)
      INTEGER I,MAXIT,DIM,M,K
      DO 5 I=1,DIM
    5 EIVEK(I) = 1.0
      EIWB = 0.0
      DO 16 M=1,MAXIT
      EIWA = EIWB
      DO 14 I=1,DIM
      HFELD(I) = 0.0
      DO 14 K=1,DIM
   14 HFELD(I) = HFELD(I) + MATRIX(I,K) * EIVEK(K)
      DO 15 I=1,DIM
   15 EIVEK(I) = HFELD(I) / HFELD(DIM)
      EIWB = HFELD(DIM)
      REST = ABS(EIWB - EIWA)
   16 CONTINUE
      RETURN
      END
```

Listing 7. Bestimmung des größten Eigenvektors nach Mises

```
Eingangs-, Ausgangsgrößen:

   MATRXA ( REAL MATRXA (DIMA) )  : Ausgangsmatrix A spaltenweise im
                                    Feld MATRXA abgelegt
   MATRXB ( REAL MATRXB (DIMB) )  : Ausgangsmatrix B spaltenweise im
                                    Feld MATRXB abgelegt
   MATRXC ( REAL MATRXC (DIMC) )  : Ergebnismatrix des Produktes der
                                    Matrizen A und B, spaltenweise
                                    abgelegt
   DIMA    ( INTEGER           )  : Dimension des Feldes MATRXA
   DIMB    ( INTEGER           )  : Dimension des Feldes MATRXB
   DIMC    ( INTEGER           )  : Dimension des Feldes MATRXC
   SPLT    ( INTEGER           )  : Spaltenzahl der Ergebnismatrix
   ZL      ( INTEGER           )  : Zeilenzahl der Ergebnismatrix

C----------------------------------.MAIN_MATPRO.-------------------
      REAL A(4),B(4),C(4)
      INTEGER I,K
      A(1)=2
      A(2)=0
      A(3)=0
      A(4)=2
      B(1)=15
      B(2)=12
      B(3)=1
      B(4)=5
      I=2
      K=2
      CALL MATPRO (A,4,B,4,C,4,I,K)
      PRINT*,C(1),C(2),C(3),C(4)
      END
```

```
C Bibliotheksprogramm Nr. 8
C*******************************************************************
    SUBROUTINE MATPRO (MATRXA,DIMA,MATRXB,DIMB,MATRXC,DIMC,SPLT,ZL)
C*******************************************************************
C  - Michael Zwenger -     2913 Apen     Am Kirchweg 22       Januar 1986
C
        INTEGER DIMC,DIMA,DIMB,SPLT,ZL
        REAL MATRXA(DIMA),MATRXB(DIMB),MATRXC(DIMC)
        L = DIMA / SPLT
        M = 1
  1     N = 1
        NL = 0
        NI = 0
  2     S = 0.0
        J = 1
        JI = 0
  3     IA = M + JI
        IB = J + NL
        S = S + MATRXA(IA) * MATRXB(IB)
        J = J + 1
        JI = JI + SPLT
        IF (J.LE.L) GOTO 3
        IA = M + NI
        MATRXC(IA) = S
        N = N + 1
        NL =NL + L
        NI = NI + SPLT
        IF(N.LE.ZL) GOTO 2
        M = M + 1
        IF(M.LE.SPLT) GOTO 1
        RETURN
        END
```

Listing 8. Produkt zweier Matrizen

```
Eingangs-, Ausgangsgrößen:

 DIM     ( INTEGER             ) : Kantenlänge der Ausgangsmatrix
 DIMQUA  ( INTEGER             ) : Elementezahl der Matrix =
                                   Quadrat von DIM
 MATRIX  ( REAL MATRIX(DIMQUA) ) : eindimensionales Feld der Länge
                                   DIMQUA, in der die Ausgangsmatrix
                                   spaltenweise abgelegt ist

C-----------------------------------.MAIN_DETERMINAnTE.-------------
        REAL A(4),E,ERGEB
        INTEGER NN,NNNN
        A(1)=1
        A(2)=2
        A(3)=3
        A(4)=4
        NN=2
        NNNN=NN*NN
        ERGEB = DETER (A,NNNN,NN)
        PRINT*,ERGEB
        END

C Bibliotheksprogramm Nr. 9
C*******************************************************************
        FUNCTION DETER (MATRIX,DIMQUA,DIM)
C*******************************************************************
C  - Michael Zwenger -     2913 Apen     Am Kirchweg 22       Januar 1986
C
        INTEGER DIM,DIMQUA
        REAL MATRIX(DIMQUA)
        NT=1
        I=1
        DETER=0.0
  1     K=(I-2)*DIM+I
        J=I
        B=0.0
  2     K=K+DIM
        H=MATRIX(K)
        IF(ABS(B).GE.ABS(H)) GOTO 3
        B=H
        I1=J
  3     J=J+1
        IF(J.LE.DIM) GOTO 2
        K=I*DIM-DIM
        IF(I1.EQ.I) GOTO 5
        L=I1*DIM-DIM
        J=I
  4     I1=K+J
        I2=L+J
        H=MATRIX(I2)
        MATRIX(I2)=MATRIX(I1)
        MATRIX(I1)=H
        J=J+1
        IF(J.LE.DIM) GOTO 4
        NT=-NT
  5     I1=K+I
  6     L=DIM*I+I
        H=MATRIX(I1+1)/B
        J=I1+1+DIM
  7     MATRIX(J)=MATRIX(J)-MATRIX(L)*H
        L=L+DIM
        J=J+DIM
        IF(J.LE.DIMQUA) GOTO 7
        I1=I1+1
        IF(I1.LT.K+DIM) GOTO 6
        I=I+1
        IF(I.LT.DIM) GOTO 1
        I=1
        DETER=NT
  8     DETER=DETER*MATRIX(I)
        I=I+DIM+1
        IF(I.LE.DIMQUA) GOTO 8
        RETURN
        END
```

Listing 9. Determinante einer Matrix

```
      SUBROUTINE BILD1(XMIN,XMAX,YMIN,YMAX,FLAG,N,XE,YE,PUNKT,AN)

C*******************************************************************
C*    Christian Träger                                             *
C*    Werner-Heisenberg-Weg 39/2c                                  *
C*    8014 Neubiberg                                               *
C*    Januar 1986                                                  *
C*******************************************************************

      IMPLICIT INTEGER (I-N)
      REAL XMIN,XMAX,YMIN,YMAX,H,XE,YE
      REAL PUNKT(99,2)
      DIMENSION WERTE(5000,2)
      DIMENSION NWERT(5000,2)
      INTEGER FLAG,HANDLE,DUMMY,AESRET,AN
      INTEGER*2 CONTRL,INTIN,INTOUT,PTSIN,PTSOUT
      INTEGER*2 WORKIN(0:10),WORKOU(0:56),XY(0:201)
      COMMON /CONTRL/ CONTRL(0:11)
      COMMON /INTIN/  INTIN(0:80)
      COMMON /INTOUT/ INTOUT(0:45)
      COMMON /PTSIN/  PTSIN(0:12)
      COMMON /PTSOUT/ PTSOUT(0:12)
C                                                   INITIALISATION
      CALL APPLIN
      CALL GRAFHA(DUMMY,DUMMY,DUMMY,DUMMY)
      HANDLE=AESRET()
      DO 1 I=0,9
    1 WORKIN(I)=1
      WORKIN(10)=2
      CALL VOPNVW(WORKIN,HANDLE,WORKOU)
      CALL VHIDEC(HANDLE)
      CALL VCLRWK(HANDLE)
C                                                   MARKER-TYP,GROESSE
      I=VSMTYP(HANDLE,5)
      I=VSMHEI(HANDLE,5)
C                                                   SCHRITTWEITE
      H=ABS(XMAX-XMIN)/N
C                                                   FUNKTIONSWERTE
      XNULL=XMIN
      YRMAX=FCN(XNULL)
      YRMIN=FCN(XNULL)
      DO 2 I=1,N
      WERTE(I,1)=XNULL
      WERTE(I,2)=FCN(XNULL)
      IF(WERTE(I,2).GT.YRMAX) YRMAX=WERTE(I,2)
      IF(WERTE(I,2).LT.YRMIN) YRMIN=WERTE(I,2)
      XNULL=XNULL+H
    2 CONTINUE
C                                                   AUTOMATIK?
      IF(FLAG.EQ.1) YMIN=YRMIN
      IF(FLAG.EQ.1) YMAX=YRMAX
C                                                   TRANSFORMATION
      DO 3 I=1,N
      XK=(WERTE(I,1)-XMIN)*639.0/ABS(XMAX-XMIN)
      YK=-(WERTE(I,2)-YMIN)*399.0/ABS(YMAX-YMIN)+399.0
      NWERT(I,1)=NINT(XK)
      NWERT(I,2)=NINT(YK)
    3 CONTINUE
C                                                   UEBERGABE AN XY
```

Listing 10. Funktionen plotten mit »BILD1«

```
      J=N/100
      K=MOD(N,100)
      XY(0)=NWERT(1,1)
      XY(1)=NWERT(1,2)
      DO 7 I=0,J-1
      DO 4 L=1,100
      XY(L*2)=NWERT(I*100+L,1)
      XY(1+L*2)=NWERT(I*100+L,2)
    4 CONTINUE
      CALL VPLINE(HANDLE,101,XY)
      XY(0)=NWERT(I*100+L,1)
      XY(1)=NWERT(I*100+L,2)
    7 CONTINUE
      L=2
      DO 5 I=(N-K),N
      XY(L)=NWERT(I,1)
      XY(L+1)=NWERT(I,2)
      L=L+2
    5 CONTINUE
      CALL VPLINE(HANDLE,K+1,XY)
C                                         BERECHNUNG ACHSENKREUZ
      ZA=1.0
      ZA=SIGN(ZA,XMIN)
      ZB=1.0
      ZB=SIGN(ZB,XMAX)
      IF(ZA.EQ.ZB) GOTO 8
      XA=-XMIN*639.0/ABS(XMAX-XMIN)
      GOTO 9
    8 XA=319
    9 ZA=1.0
      ZA=SIGN(ZA,YMIN)
      ZB=1.0
      ZB=SIGN(ZB,YMAX)
      IF(ZA.EQ.ZB) GOTO 10
      YA=YMIN*399.0/ABS(YMAX-YMIN)+399.0
      GOTO 11
   10 YA=199
   11 XY(0)=0
C                                         ZEICHNEN ACHSENKREUZ
      XY(1)=NINT(YA)
      XY(2)=639
      XY(3)=NINT(YA)
      CALL VPLINE(HANDLE,2,XY)
      XY(0)=NINT(XA)
      XY(1)=0
      XY(2)=NINT(XA)
      XY(3)=399
      CALL VPLINE(HANDLE,2,XY)
C                                         EINHEITEN ZEICHNEN
      XS=XMIN
   12 XT=(XS-XMIN)*639.0/ABS(XMAX-XMIN)
      XY(0)=NINT(XT)
      XY(1)=NINT(YA)+3
      XY(2)=NINT(XT)
      XY(3)=NINT(YA)-3
      CALL VPLINE(HANDLE,2,XY)
      XS=XS+XE
      IF(XS.LE.XMAX) GOTO 12
      YS=YMIN
   13 YT=-(YS-YMIN)*399.0/ABS(YMAX-YMIN)+399.0
      XY(0)=NINT(XA)+3
      XY(1)=NINT(YT)
      XY(2)=NINT(XA)-3
      XY(3)=NINT(YT)
      CALL VPLINE(HANDLE,2,XY)
      YS=YS+YE
      IF(YS.LE.YMAX) GOTO 13
C                                         PUNKTE ZEICHNEN
      J=AN/100
      K=MOD(AN,100)
      IF(J.EQ.0) GOTO 16
      DO 14 L=0,J-1
      DO 14 I=1,100
      XY(-2+2*I)=NINT((PUNKT(L*100+I,1)-XMIN)*639.0/ABS(XMAX-XMIN))
      XY(-1+2*I)=NINT(-(PUNKT(L*100+I,2)-YMIN)*399.0/ABS(YMAX-YMIN)
     +399.0*)
   14 CONTINUE
      CALL VPMARK(HANDLE,100,XY)
   16 J=0
      DO 15 I=(AN-K+1),AN
      XY(J)=NINT((PUNKT(I,1)-XMIN)*639.0/ABS(XMAX-XMIN))
      XY(J+1)=NINT(-(PUNKT(I,2)-YMIN)*399.0/ABS(YMAX-YMIN)+399.0)
      J=J+2
   15 CONTINUE
      CALL VPMARK(HANDLE,K,XY)
C                                         TASTENDRUCK ABWARTEN
      CALL EVNTKE
      CALL VCLSVW(HANDLE)
      END
```

Listing 10. Funktionen plotten mit »BILD1« (Schluß)

```
C     PROGRAM FOURIER
C*****************************************************************
C*    Christian Träger                                           *
C*    Werner-Heisenberg-Weg 39/2c                                *
C*    8014 Neubiberg                                             *
C*    Januar 1986                                                *
C*****************************************************************

      IMPLICIT INTEGER (I-N)
      INTEGER P,N
      REAL PUNKT(99,2),PUNKTE(99,2)
      REAL A(50),B(50)
      COMMON/KOEFF/A,B,N
      PARAMETER(PI=3.141592654)
      PRINT*,'              Fourier-Analyse'
      PRINT*
      PRINT*,' Anzahl der Stützstellen:'
      PRINT*,' N < 100, ungerade!  '
      READ(*,1) NE
    1 FORMAT(I2)
      N=(NE-1)/2
      C1=COS(2.0*PI/REAL(NE))
      S1=SIN(2.0*PI/REAL(NE))
      C=1.0
      S=0.0
      P=1
      XE=0.0
      DO 4 I=1,NE
      IA=I-1
      WRITE(*,2) IA,XE
    2 FORMAT(' Stützstelle ',I2,'  X=',F11.8)
      PRINT*
      PUNKT(I,1)=XE
C                                   Eingabe der Stützstellen
      READ(*,*) PUNKT(I,2)
      XE=REAL(I)*PI*2.0/REAL(NE)
    4 CONTINUE
C                                   Beginn des Algorithmus
    5 U2=0
      U1=0
      NN=2*N+1
    6 U0=PUNKT(NN+1,2)+2*C*U1-U2
      U2=U1
      U1=U0
      NN=NN-1
      IF(NN.NE.1) GOTO 6
      A(P)=(2.0/REAL(NE))*(PUNKT(1,2)+C*U1-U2)
      B(P)=(2.0/REAL(NE))*S*U1
      IF(P.EQ.N+1) GOTO 7
      Q=C1*C-S1*S
      S=C1*S+S1*C
      C=Q
      P=P+1
      GOTO 5
    7 DO 8 I=1,NE
      PUNKTE(I,1)=PUNKT(I,1)
      PUNKTE(I,2)=PUNKT(I,2)
    8 PRINT*, PUNKTE(I,2)
      N1=NE
C                               Ausgabe
      CALL BILD1(-2.0*PI,2.0*PI,-2.0,2.0,0,999,PI/2.0,0.1,PUNKTE
     *,N1)
      END
C                               Fourierreihe
      FUNCTION FCN(X)
      COMMON/KOEFF/A(50),B(50),N
      FCN=0.5*A(1)
      DO 1 I=2,N+1
      FCN=FCN+A(I)*COS(REAL(I-1)*X)
      FCN=FCN+B(I)*SIN(REAL(I-1)*X)
    1 CONTINUE
      RETURN
      END

LINKER - CONTROL File zum Linken des Programmes BILD1:

;
;   Linker-File for linking Pro Fortran-77 Programs under GEMDOS
;   using the F77-GEM interface library and for linking the
;   SUBROUTINE BILD1
;
INPUT PLINIT
INPUT *
INPUT BILD1
LIBRARY F77GEM
LIBRARY F77LIB
INPUT PLEND
DATA 4K
COMMON DUMMY
```

Listing 11. Fourieranalyse

Drucker-Einstellung leichtgemacht

Accessories sind ein großer Pluspunkt des GEM-Desktop. Mit unserer Hilfe programmieren Sie Ihre eigenen Accessories.

Vorbei ist die Zeit des Handbuchwälzens, Steuerzeichensuchens und der abgebrochenen Dipschalter. Alle gewünschten Vorgaben stellen Sie einfach über das hier beschriebene Drucker-Accessory ein.

Da aber viele ST-Besitzer selbst gute Ideen für Accessories besitzen, aber mit der Problematik bei der Programmierung Schwierigkeit haben, ist die Beschreibung besonders ausführlich ausgefallen und dient damit als ausgezeichnete Vorlage für eigene Accessories.

Anwendung:

Das Accessory wird auf die System- oder Startdiskette kopiert und anschließend neu gebootet. Nur durch einen Boot-Vorgang lassen sich Accessories in das Desktop einbinden und anschließend auch durch die Maus ansprechen. Während der Initialisierung des Drucker-Accessorys wird ein auf »On Line« geschalteter Drucker mit der durch das Programm vorgegebenen Grundeinstellung initialisiert. Eine erfolgreiche Anpassung quittiert der Drucker mit einem Piepser. Ist der Drucker ausgeschaltet oder nicht empfangsbereit, setzt das Betriebssystem den Boot-Vorgang fort, ohne daß Druckercodes an die parallele Schnittstelle ausgegeben werden.

Nachdem sich das Betriebssystem mit dem Desktop zurückgemeldet hat, kann man das Accessory als Eintrag im Desktop unter dem Atari-Symbol finden und durch einen Mausklick starten. Die Eingabemaske der dann erscheinenden Dialogbox erklärt sich weitgehend von selbst. Mit dem Print-Schalter überprüft man die gewählte Druckereinstellung. Als Test bringt der Drucker die ASCII-Zeichen von 32 bis 127 zu Papier.

Stellt das Programm beim Anklicken des »OK«-Feldes mit der Maus fest, daß der Drucker nicht empfangsbereit ist, erscheint ein Fenster, das auf diesen Fehler hinweist. Eine erfolgreiche Initialisierung quittiert der Drucker wieder mit einem Piepser. Bitte beachten Sie: Wenn ein Druckerspooler aktiviert wurde, kann das Accessory nicht feststellen, ob der Drucker empfangsbereit ist, da der Spooler die Daten aufnimmt und die Rückmeldung ausführt. Anschließend schließt sich die Dialogbox. Bei einem erneuten Aufruf des Accessory erscheint die zuletzt gewählte Maskeneinstellung.

Die Anpassung an andere Druckertypen fällt nicht schwer. Sie beschränkt sich im wesentlichen auf die Änderung der Druckerkontrollcodes in der Druckertabelle, der Namensänderung im Menü-Eintrag und, falls man darauf Wert legt, in der Einstellung der Grundmaske der Dialogbox.

Druckertabelle:

Variablennamen dürfen nicht gelöscht werden, wenn entsprechende Druckerfunktionen fehlen. Deren Inhalt ist lediglich durch $FF zu ersetzen.

Überschrift der Dialogbox:

Die Überschrift der Dialogbox kann man beliebig ändern. Nur die Anzahl der Buchstaben darf in keinem Fall unter- oder überschritten werden! Falls der Name kürzer ist, dienen Leerzeichen als Platzhalter.

Einstellung der Grundmaske:

Je nach Verwendungszweck erstellt man eine spezielle Grundmaske, zum Beispiel mit deutschem statt amerikanischem Zeichensatz. Dabei sind zu unterscheiden:

EIN/AUS-Tasten

Diese Tasten werden mit der Funktion
`select_obj(Objekt ii,SELECTED);`
voreingestellt. Sollen Tasten als nicht benutzbar dargestellt werden findet die Funktion
`do_obj(Objekt ii,DISABLED);`
Anwendung.

Zahlenfelder:

Die Anfangswerte der Zahlenfelder liegen in den entsprechenden Druckercodes fest, zum Beispiel 0 Zeichen für linken Druckrand in »p__lmar«. Gleiches gilt für Maximal- und Minimalwerte der veränderlichen Werte.

Textfeld:

Das Textfeld für den internationalen Zeichensatz kann in seinem Startwert in
`p_set[] = { 27, 82, 2, 255 };`
verändert werden, zum Beispiel für den deutschen Zeichensatz.

(Wolfram Winter/hb)

Das Listing finden Sie auf Seite 142.

Plädoyer für eine Nebensache

Accessories sind das Salz in der Suppe des Desktop. Welcher ST-Besitzer legt schon Wert darauf, die vielfältigen Funktionen, die ihm die Accessories bieten, auf herkömmlichem Wege mit Steuercodes einzustellen. Erst einmal im Handbuch wühlen und die richtigen Codes zu finden, anschließend das gerade laufende Programm unterbrechen, um überhaupt Steuercodes absenden zu können? Wie einfach ist dagegen doch das Ändern der Baudrate, der Uhrzeit oder der kompletten Druckeranpassung über die Accessories!

Na, das wissen wir ST-Fans doch schon lange, und das schätzen wir alle sehr. Warum dann dieses kleine Plädoyer für Accessories?

Es ist überflüssig, meinen Sie? Nein, denn die Softwarehäuser machen nicht mehr mit! Und das aus gutem Grund. Die Entwicklung kostet Zeit und somit Geld. Ein Accessory auf einer Diskette mit Kopierschutz zu vertreiben, ist sinnlos. Bei einer kopiergeschützten Diskette kann man sich noch helfen, indem man die anderen Accessories auf diese Diskette kopiert. Was macht man aber bei mehreren Accessories? Jedes auf einer kopiergeschützten Diskette? Da ist guter Rat teuer. Ein Accessory wird ausschließlich durch den Bootvorgang initialisiert.

Kopiergeschützte Disketten bringen weitere Probleme: Jeder Boot-Vorgang beansprucht diese Diskette aufs neue. Die Diskette verschleißt, produziert Lesefehler, und vorbei ist es mit dem schönen Accessory.

Läßt man den Kopierschutz weg, passiert genau dasselbe wie mit dem »Programmers Calculator« von GTS aus Berlin. Er avancierte zur Raubkopie mit der höchsten Verbreitung. Dazu folgende Anekdote:

Auf der CeBIT wurde Manuel Drösler, der Geschäftsführer von GTS, vom Mitarbeiter eines amerikanischen Softwarehauses angesprochen, ob er nicht gerne einen erstklassigen Rechner als Accessory hätte. Nach dem Booten des Betriebssystems machte alles große Augen. Überrascht stellte Drösler fest, daß es sich um sein Produkt handelte. Bis nach Amerika war also sein Ruf als guter Programmierer bereits gedrungen! So kann man unfreiwillig eine Bestätigung für die gute Qualität seiner Produkte bekommen.

Aber diese kleine Geschichte macht wohl jedem klar, daß die Softwarehäuser verständlicherweise nicht mehr mitspielen, weiterhin neue und immer bessere Accessories zu entwickeln. Nein, da müssen die ST-Hobby-Programmierer zur Selbsthilfe greifen. Aber Accessories haben auch ihre Tücken. Ob die ST-Gemeinde pfiffig genug ist und das in den Griff bekommt, werden Sie in der nächsten Ausgabe lesen. Denn hiermit fordern wir alle ST-Besitzer auf, ihr Können unter Beweis zu stellen. Eine gute Idee, ein guter Compiler und unsere detaillierte Veröffentlichung »Accessory« sind das Startkapital. Hoffentlich haben bald alle eine Sammlung von hilfreichen und frei kopierbaren Hilfsprogrammen für den Desktop. Ein gutes Honorar und Ihr Name im 68000er sind uns die besten Leserlistings wert.

Es wird sich zeigen, ob Jim Tittsler, Softwarespezialist bei Atari, recht hatte, als er auf meine Frage, was er von den deutschen Programmieren halte, antwortete: »Ich finde, sie haben beim ST gezeigt, daß sie sich vor niemanden verstecken brauchen.«

Wenn das kein Ansporn ist!

Unsere Adresse für Ihre Listings:
Redaktion Happy-Computer
Markt&Technik Verlag AG
»Accessory«
Hans-Pinsel-Str. 2
D-8013 Haar

(hb)

```
/**************************************************************************/
/**************************************************************************/
/****  DRUCKER-ACCESSORY FUER BELIEBIGE DRUCKERTYPEN      Version 4.00 ****/
/****  ----------------------------------------------                 ****/
/****  Druckertyp : EPSON FX-80                                       ****/
/****  Stand      : 20.06.86                                          ****/
/****                                                                 ****/
/****  Programmautor : Wolfram Winter                                 ****/
/****                  Eupenerstr. 5                                  ****/
/****                  D5600 Wuppertal 11                             ****/
/**************************************************************************/
/**************************************************************************/
/*------------------------------------------------------------------------*/
/*     DEFINES fuer DR-Compiler (bei Bedarf fuer andere Compiler aendern) */
/*------------------------------------------------------------------------*/
#define BYTE        char
#define WORD        int                 /* LATTICE-Compiler: short          */
#define UWORD       unsigned int        /* LATTICE-Compiler: unsigned short */
#define LONG        long
#define EXTERN      extern
#define VOID        /**/
/*------------------------------------------------------------------------*/
/*      Vereinbarungen                                                    */
/*------------------------------------------------------------------------*/
EXTERN      gl_apid;                    /* Applikationskennung            */
EXTERN LONG gemdos();                   /* GEMDOS-Funktion                */
#define  Cprnos()     gemdos(0x11)      /* Drucker-Anwesenheit            */
#define  Cprnout(a)   gemdos(0x5,a)     /* Drucker-Ausgabe                */
/*------------------------------------------------------------------------*/
WORD   contrl[12];                      /* Array der Kontrollparameter    */
WORD   intin[128];                      /* Array d. Integer-Eingabeparameter*/
WORD   ptsin[128];                      /* Array d. Eingabe-Koordinaten   */
WORD   intout[128];                     /* Array d. Integer-Ausgabeparameter*/
WORD   ptsout[128];                     /* Array d. Ausgabe-Koordinaten   */
WORD   work_in[11];                     /* Eingabe zum GSX-Array          */
WORD   work_out[57];                    /* Ausgabe vom GSX-Array          */
WORD   phys_handle;                     /* Desk-Handle                    */
/*------------------------------------------------------------------------*/
WORD   bchar;                           /* Breite u. Hoehe einer Buchstaben-*/
```

```
WORD  hchar;                                /* zelle, quadratische Box um eine */
WORD  bbox;                                 /* Buchstabenzelle in Breite und   */
WORD  hbox;                                 /* Hoehe                           */
/*-----------------------------------------------------------------------------*/
WORD  menu_id;                              /* Nummer des Menue-Eintrages      */
WORD  ret;                                  /* Dummy                           */
LONG  tree;                                 /* Adresse Baumstruktur            */
WORD  event;                                /* Ereignis                        */
WORD  msgbuff[8];                           /* Nachrichtenpuffer               */
/*-----------------------------------------------------------------------------*/
typedef struct object                       /* OBJECT-Struktur                 */
{                                           /*                                 */
WORD    ob_next;                            /* naechste Objekt-Ebene           */
WORD    ob_head;                            /* Zeiger auf Anfang der Objekte   */
WORD    ob_tail;                            /* Zeiger auf Ende der Objekte     */
UWORD   ob_type;                            /* Objekt-Typ                      */
UWORD   ob_flags;                           /* Objekt-Flag                     */
UWORD   ob_state;                           /* Objekt-Status                   */
LONG    ob_spec;                            /* Zeiger Objektspezifikation      */
WORD    ob_x;                               /* X-Koordinate Objekt             */
WORD    ob_y;                               /* Y-Koordinate Objekt             */
WORD    ob_width;                           /* Breite Objekt                   */
WORD    ob_height;                          /* Hoehe Objekt                    */
} OBJECT;                                   /*                                 */
                                            /*                                 */
typedef struct grect                        /* GRECT-Struktur                  */
{                                           /*                                 */
WORD    g_x;                                /* X-Koordinate                    */
WORD    g_y;                                /* Y-Koordinate                    */
WORD    g_w;                                /* Breite                          */
WORD    g_h;                                /* Hoehe                           */
} GRECT;                                    /*                                 */
                                            /*                                 */
typedef struct text_edinfo                  /* TEDINFO-Struktur                */
{                                           /*                                 */
LONG    te_ptext;                           /* Zeiger auf aktuellen Text       */
LONG    te_ptmplt;                          /* Zeiger auf Text-Maske           */
LONG    te_pvalid;                          /* Zeiger auf erlaubte Eingabewerte */
WORD    te_font;                            /* Schriftgroesse                  */
WORD    te_junk1;                           /* nicht benutzt                   */
WORD    te_just;                            /* Text-Justierung                 */
WORD    te_color;                           /* Text-Farbe                      */
WORD    te_junk2;                           /* nicht benutzt                   */
WORD    te_thickness;                       /* Dicke der Box-Umrandung         */
WORD    te_txtlen;                          /* Text-Laenge                     */
WORD    te_tmplen;                          /* Masken-Laenge                   */
} TEDINFO;                                  /*                                 */
/*-----------------------------------------------------------------------------*/
#define X_BAK           0x0100              /* TOUCHEXIT-Pfeil (links)         */
#define X_FWD           0x0200              /* TOUCHEXIT-Pfeil (rechts)        */
#define NIL             -1                  /*                                 */
#define FOREVER         for(;;)             /* Endlos-Schleife                 */
#define YES             1                   /*                                 */
#define NO              0                   /*                                 */
#define MU_MESAG        0x0010              /* Nachricht fuer event            */
#define AC_OPEN         40                  /* Accessory geoeffnet?            */
#define BEG_UPDATE      1                   /* Benutzeraktivitaeten            */
#define END_UPDATE      0                   /*                                 */
/*-----------------------------------------------------------------------------*/
#define ROOT            0                   /* Objekt-Formen                   */
#define G_BOX           20
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker

```
#define G_TEXT          21
#define G_BOXTEXT       22
#define G_IMAGE         23
#define G_USERDEF       24
#define G_IBOX          25
#define G_BUTTON        26
#define G_BOXCHAR       27
#define G_STRING        28
#define G_FTEXT         29
#define G_FBOXTEXT      30
#define G_ICON          31
#define G_TITLE         32
#define NONE            0x0
#define SELECTABLE      0x1
#define DEFAULT         0x2
#define EXIT            0x4
#define EDITABLE        0x8
#define RBUTTON         0x10
#define LASTOB          0x20
#define TOUCHEXIT       0x40
#define HIDETREE        0x80
#define INDIRECT        0x100
#define NORMAL          0x0
#define SELECTED        0x1
#define CROSSED         0x2
#define CHECKED         0x4
#define DISABLED        0x8
#define OUTLINED        0x10
#define SHADOWED        0x20
/*-----------------------------------------------------------------------------*/
#define OB_NEXT(x)      (tree + (x) * sizeof(OBJECT) + 0)   /* Objekt-Attribute */
#define OB_HEAD(x)      (tree + (x) * sizeof(OBJECT) + 2)
#define OB_TAIL(x)      (tree + (x) * sizeof(OBJECT) + 4)
#define OB_TYPE(x)      (tree + (x) * sizeof(OBJECT) + 6)
#define OB_FLAGS(x)     (tree + (x) * sizeof(OBJECT) + 8)
#define OB_STATE(x)     (tree + (x) * sizeof(OBJECT) + 10)
#define OB_SPEC(x)      (tree + (x) * sizeof(OBJECT) + 12)
#define OB_X(x)         (tree + (x) * sizeof(OBJECT) + 16)
#define OB_Y(x)         (tree + (x) * sizeof(OBJECT) + 18)
#define OB_WIDTH(x)     (tree + (x) * sizeof(OBJECT) + 20)
#define OB_HEIGHT(x)    (tree + (x) * sizeof(OBJECT) + 22)
/*-----------------------------------------------------------------------------*/
#define LWGET(x)        ( (WORD) *((WORD *)(x)) )   /* get WORD pointed by WORD */
#define LWSET(x, y)     ( *((WORD *)(x)) = y)       /* set WORD pointed by WORD */
#define LLGET(x)        ( *((LONG *)(x)))           /* get LONG pointed by LONG */
#define LLSET(x, y)     ( *((LONG *)(x)) = y)       /* set LONG pointed by LONG */
/*-----------------------------------------------------------------------------*/
/*       Drucker-Codes und Menue-Eintrag                                       */
/*-----------------------------------------------------------------------------*/
#define MENU_NAME  "  FX-80 Drucker"                /* Name Menue-Eintrag      */
WORD p_reset[]     = { 27, 64,255 };                /* Drucker-Reset           */
WORD p_unidir[]    = { 27, 85,  1,255 };            /* Druck von einer Seite   */
WORD p_bidir[]     = { 27, 85,  0,255 };            /* Druck von beiden Seiten */
WORD p_set[]       = { 27, 82,  0,255 };            /* Zeichensatz             */
#define  MIN_SET  0
#define  MAX_SET  7
BYTE  *str_set[]   = { "USA","F","D","GB","DK","S","I","S" };
WORD p_page[]      = { 27, 67,  0, 12,255 };        /* 12 Zoll-Seite           */
#define  MIN_PAGE 1
#define  MAX_PAGE 22
WORD p_skip[]      = { 27, 78,  6,255 };            /* unten 6Z. ueberspringen */
WORD p_offskip[]   = { 27, 79,255 };                /* kein Perforationssprung */
#define  MIN_SKIP 0
```

```
#define  MAX_SKIP 127
WORD p_lmar[]     = { 27,108,  0,255 };               /* 0Zeichen linker Rand      */
#define  MIN_LMAR 0
WORD p_rmar[]     = { 27, 81,135,255 };               /* 135Zeichen rechter Rand */
#define  MAX_RMAR 135
WORD p_lspace[]   = { 27, 51, 36,255 };               /* 36/216 Zeilenabstand      */
#define  MIN_LSPA 0
#define  MAX_LSPA 255
WORD p_stand[]    = { 27, 53,               255 };  /*           STANDARD an     */
WORD p_nlq[]      = { 27, 53,               255 };  /* STANDARD an                */
WORD p_ital[]     = { 27, 52,               255 };  /*             ITALIC an     */
WORD p_pica[]     = { 27, 84, 27, 80, 18,255 };  /* SUP,ELIT,COND aus         */
WORD p_cond[]     = { 27, 84, 27, 80, 15,255 };  /* SUP,ELIT aus,COND an      */
WORD p_elite[]    = { 27, 84, 18, 27, 77,255 };  /* SUP,COND aus,ELIT an      */
WORD p_micro[]    = { 27, 80, 18, 27, 83,0,255}; /* ELIT,COND aus,SUP an      */
WORD p_onexp[]    = { 27, 87,  1,255 };               /* EXPAND an                 */
WORD p_offexp[]   = { 27, 87,  0,255 };               /* EXPAND aus                */
WORD p_onprop[]   = {           255 };                /* PROPORTIONAL an           */
WORD p_offprop[]  = {           255 };                /* PROPORTIONAL aus          */
WORD p_ondoub[]   = { 27, 71,255 };                   /* DOPPEL an                 */
WORD p_offdoub[]  = { 27, 72,255 };                   /* DOPPEL aus                */
WORD p_onemph[]   = { 27, 69,255 };                   /* EMPHASIZED an             */
WORD p_offemph[]  = { 27, 70,255 };                   /* EMPHASIZED aus            */
WORD p_onzero[]   = {           255 };                /* durchstrichene Null       */
WORD p_offzero[]  = {           255 };                /* normale Null              */
WORD p_onform[]   = { 12,255 };                       /* Seitenvorschub            */
WORD p_ret[]      = {  7, 13,255 };                   /* Klingel, Return           */
/*-------------------------------------------------------------------------------*/
/*                                                                               */
/*       Definitionen aus RSC-Set                                                */
/*-------------------------------------------------------------------------------*/
BYTE *rs_strings[] = {
"   EPSON FX80 DRUCKER INITIALISIERUNG   ","","",
" (c) Wolfram Winter  D5600 WUPPERTAL-11 ","","",
"DRUCK","","","->","<->","SEITE","","","12","","","'Länge","","",
"ZEICHEN","","","UMBRUCH","",""," 6","","","?","ABSTAND","",""," 36","","",
"/216'","","","PROPORT","","","AN","AUS","LI.RAND","",""," 0","","",
"EXPAND","","","AN","AUS","RE.RAND","","","137","","","DOUBLE","","",
"AN","AUS","EMPH","","","AN","AUS"," AUSGABE MIT: ","","","NULL","","",
"0","O","RESET","","","AN","AUS","FORMFEED","","","AN","AUS","PICA","MICRO",
"ELITE","CONDEN","STAND","NLQ","ITALIC","PRINT","","","AN","AUS","ABBRUCH",
"OK","Version 4.0 ","",""};

TEDINFO rs_tedinfo[] = {
  0L,   1L,   2L, 3, 6, 2, 0x11E1, 0x0, -1, 41,1,
  3L,   4L,   5L, 5, 6, 0, 0x1180, 0x0, -1, 41,1,
  6L,   7L,   8L, 3, 6, 0, 0x1180, 0x0, -1,  6,1,
 11L,  12L,  13L, 3, 6, 0, 0x1180, 0x0, -1,  6,1,
 14L,  15L,  16L, 3, 6, 2, 0x1180, 0x0, -1,  3,1,
 17L,  18L,  19L, 3, 6, 0, 0x1180, 0x0, -1,  7,1,
 20L,  21L,  22L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 23L,  24L,  25L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 26L,  27L,  28L, 3, 6, 2, 0x1180, 0x0, -1,  4,1,
 30L,  31L,  32L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 33L,  34L,  35L, 3, 6, 2, 0x1180, 0x0, -1,  4,1,
 36L,  37L,  38L, 3, 6, 0, 0x1180, 0x0, -1,  6,1,
 39L,  40L,  41L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 44L,  45L,  46L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 47L,  48L,  49L, 3, 6, 2, 0x1180, 0x0, -1,  4,1,
 50L,  51L,  52L, 3, 6, 0, 0x1180, 0x0, -1,  7,1,
 55L,  56L,  57L, 3, 6, 0, 0x1180, 0x0, -1,  8,1,
 58L,  59L,  60L, 3, 6, 2, 0x1180, 0x0, -1,  4,1,
 61L,  62L,  63L, 3, 6, 0, 0x1180, 0x0, -1,  7,1,
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Fortsetzung)

```
 66L,  67L,  68L, 3, 6, 0, 0x1180, 0x0, -1,  5,1,
 71L,  72L,  73L, 3, 6, 2, 0x11A1, 0x0, -1, 15,1,
 74L,  75L,  76L, 3, 6, 0, 0x1180, 0x0, -1,  5,1,
 79L,  80L,  81L, 3, 6, 0, 0x1180, 0x0, -1,  6,1,
 84L,  85L,  86L, 3, 6, 0, 0x1180, 0x0, -1,  9,1,
 96L,  97L,  98L, 3, 6, 0, 0x1180, 0x0, -1,  6,1,
103L, 104L, 105L, 5, 6, 2, 0x1180, 0x0, -1, 13,1 };

OBJECT rs_object[] = {
-1,  1, 81, G_BOX,      NONE,       SHADOWED, 0x21100L    0,  0, 47,  17,
 2, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x0L,       0,  0, 47,   2,
 3, -1, -1, G_TEXT,     NONE,       NORMAL,   0x1L,       9,  2, 30,1536,
 4, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x2L,       1,  3,  8,   1,
 7,  5,  6, G_BOX,      NONE,       NORMAL,   0xFF0121L, 10,  3, 13,   1,
 6, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x9L,       0,  0,  6,   1,
 4, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0xAL,       7,  0,  6,   1,
 8, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x3L,      24,  3,  8,   1,
13,  9, 12, G_IBOX,     NONE,       NORMAL,   0xFF1100L, 33,  3, 13,   1,
10, -1, -1, G_BOXTEXT,  NONE,       NORMAL,   0x4L,       2,  0,  3,   1,
11, -1, -1, G_TEXT,     NONE,       NORMAL,   0x5L,       5,  0,  6,   1,
12, -1, -1, 0x21B,      TOUCHEXIT,  SHADOWED, 0x3FF1100L, 11, 0,  2,   1,
 8, -1, -1, 0x11B,      TOUCHEXIT,  NORMAL,   0x4FF1100L, 0,  0,  2,   1,
14, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x6L,       1,  4,  8,   1,
15, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x7L,      24,  4,  8,   1,
19, 16, 18, G_IBOX,     NONE,       NORMAL,   0xFF1100L, 33,  4, 13,   1,
17, -1, -1, G_BOXTEXT,  NONE,       NORMAL,   0x8L,       2,  0,  9,   1,
18, -1, -1, 0x21B,      TOUCHEXIT,  SHADOWED, 0x3FF1100L, 11, 0,  2,   1,
15, -1, -1, 0x11B,      TOUCHEXIT,  NORMAL,   0x4FF1100L, 0,  0,  2,   1,
21, 20, 20, G_BOX,      NONE,       NORMAL,   0xFF0100L, 10,  4, 13,   1,
19, -1, -1, G_BUTTON,   TOUCHEXIT,  SHADOWED, 0x1DL,      0,  0, 13,   1,
22, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x9L,      24,  5,  8,   1,
27, 23, 26, G_IBOX,     NONE,       NORMAL,   0xFF1100L, 33,  5, 13,   1,
24, -1, -1, G_BOXTEXT,  NONE,       NORMAL,   0xAL,       2,  0,  4,   1,
25, -1, -1, G_TEXT,     NONE,       NORMAL,   0xBL,       6,  0,  5,   1,
26, -1, -1, 0x21B,      TOUCHEXIT,  SHADOWED, 0x3FF1100L, 11, 0,  2,   1,
22, -1, -1, 0x11B,      TOUCHEXIT,  NORMAL,   0x4FF1100L, 0,  0,  2,   1,
28, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0xCL,       1,  6,  8,   1,
31, 29, 30, G_BOX,      NONE,       NORMAL,   0xFF0121L, 10,  6, 13,   1,
30, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x2AL,      0,  0,  6,   1,
28, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x2BL,      7,  0,  6,   1,
32, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0xDL,      24,  6,  8,   1,
36, 33, 35, G_IBOX,     NONE,       NORMAL,   0xFF1100L, 33,  6, 13,   1,
34, -1, -1, G_BOXTEXT,  NONE,       NORMAL,   0xEL,       2,  0,  9,   1,
35, -1, -1, 0x21B,      TOUCHEXIT,  SHADOWED, 0x3FF1100L, 11, 0,  2,   1,
32, -1, -1, 0x11B,      TOUCHEXIT,  NORMAL,   0x4FF1100L, 0,  0,  2,   1,
37, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0xFL,       1,  7,  8,   1,
40, 38, 39, G_BOX,      NONE,       NORMAL,   0xFF0121L, 10,  7, 13,   1,
39, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x35L,      0,  0,  6,   1,
37, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x36L,      7,  0,  6,   1,
41, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x10L,     24,  7,  8,   1,
45, 42, 44, G_IBOX,     NONE,       SHADOWED, 0xFF1100L, 33,  7, 13,   1,
43, -1, -1, G_BOXTEXT,  NONE,       NORMAL,   0x11L,      2,  0,  9,   1,
44, -1, -1, 0x21B,      TOUCHEXIT,  NORMAL,   0x3FF1100L, 11, 0,  2,   1,
41, -1, -1, 0x11B,      TOUCHEXIT,  NORMAL,   0x4FF1100L, 0,  0,  2,   1,
46, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x12L,      1,  8,  8,   1,
49, 47, 48, G_BOX,      NONE,       NORMAL,   0xFF0121L, 10,  8, 13,   1,
48, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x40L,      0,  0,  6,   1,
46, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x41L,      7,  0,  6,   1,
50, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x13L,      1,  9,  8,   1,
53, 51, 52, G_BOX,      NONE,       NORMAL,   0xFF0121L, 10,  9, 13,   1,
52, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x45L,      0,  0,  6,   1,
50, -1, -1, G_BUTTON,   0x11,       SHADOWED, 0x46L,      7,  0,  6,   1,
54, -1, -1, G_BOXTEXT,  NONE,       SHADOWED, 0x14L,     24,  9, 22,   1,
```

```
 55, -1, -1, G_BOXTEXT, NONE,    SHADOWED, 0x15L,       1, 10,  8,    1,
 58, 56, 57, G_BOX,     NONE,    NORMAL,   0xFF0121L,  10, 10, 13,    1,
 57, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x4DL,       0,  0,  6,    1,
 55, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x4EL,       7,  0,  6,    1,
 59, -1, -1, G_BOXTEXT, NONE,    SHADOWED, 0x16L,      24, 10,  8,    1,
 62, 60, 61, G_BOX,     NONE,    NORMAL,   0xFF0121L,  33, 10, 13,    1,
 61, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x52L,       0,  0,  6,    1,
 59, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x53L,       7,  0,  6,    1,
 63, -1, -1, G_BOXTEXT, NONE,    SHADOWED, 0x17L,      24, 11,  8,    1,
 66, 64, 65, G_BOX,     NONE,    NORMAL,   0xFF0121L,  33, 11, 13,    1,
 65, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x57L,       0,  0,  6,    1,
 63, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x58L,       7,  0,  6,    1,
 71, 67, 70, G_BOX,     NONE,    NORMAL,   0xFF0100L,   2, 12,  9,    4,
 68, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x59L,       0,  0,  9,    1,
 69, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5AL,       0,  1,  9,    1,
 70, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5BL,       0,  2,  9,    1,
 66, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5CL,       0,  3,  9,    1,
 75, 72, 74, G_BOX,     NONE,    NORMAL,   0xFF0100L,  13, 12,  9,    3,
 73, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5DL,       0,  0,  9,    1,
 74, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5EL,       0,  1,  9,    1,
 71, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x5FL,       0,  2,  9,    1,
 76, -1, -1, G_BOXTEXT, NONE,    SHADOWED, 0x18L,      24, 12,  8,    1,
 79, 77, 78, G_BOX,     NONE,    NORMAL,   0xFF0121L,  33, 12, 13,    1,
 78, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x63L,       0,  0,  6,    1,
 76, -1, -1, G_BUTTON,  0x11,    SHADOWED, 0x64L,       7,  0,  6,    1,
 80, -1, -1, G_BUTTON,  0x15,    SHADOWED, 0x65L,      25, 14,  9,    2,
 81, -1, -1, G_BUTTON,  0x17,    SHADOWED, 0x66L,      36, 14,  9,    2,
  0, -1, -1, G_TEXT,    LASTOB,  NORMAL,   0x19L,      12, 16,  9,1536 };

#define NUM_TI    26
#define NUM_OBS   82

#define DRUCKER   0    /* TREE */
#define STANDON   72   /* OBJECT in TREE #0 */
#define CANCEL    79   /* OBJECT in TREE #0 */
#define OKAY      80   /* OBJECT in TREE #0 */
#define NLQON     73   /* OBJECT in TREE #0 */
#define ITALON    74   /* OBJECT in TREE #0 */
#define PROPON    29   /* OBJECT in TREE #0 */
#define PROPOFF   30   /* OBJECT in TREE #0 */
#define RESON     60   /* OBJECT in TREE #0 */
#define FORMON    64   /* OBJECT in TREE #0 */
#define RESOFF    61   /* OBJECT in TREE #0 */
#define FORMOFF   65   /* OBJECT in TREE #0 */
#define UNIDIR    5    /* OBJECT in TREE #0 */
#define BIDIR     6    /* OBJECT in TREE #0 */
#define IMAGE     2    /* OBJECT in TREE #0 */
#define EXPON     38   /* OBJECT in TREE #0 */
#define EXPOFF    39   /* OBJECT in TREE #0 */
#define DOUBON    47   /* OBJECT in TREE #0 */
#define DOUBOFF   48   /* OBJECT in TREE #0 */
#define EMPHON    51   /* OBJECT in TREE #0 */
#define EMPHOFF   52   /* OBJECT in TREE #0 */
#define ZEROON    56   /* OBJECT in TREE #0 */
#define ZEROOFF   57   /* OBJECT in TREE #0 */
#define PICAON    67   /* OBJECT in TREE #0 */
#define MICROON   68   /* OBJECT in TREE #0 */
#define ELITEON   69   /* OBJECT in TREE #0 */
#define CONDON    70   /* OBJECT in TREE #0 */
#define PAGE      9    /* OBJECT in TREE #0 */
#define PRINTON   77   /* OBJECT in TREE #0 */
#define PRINTOFF  78   /* OBJECT in TREE #0 */
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Fortsetzung)

```
#define LSPACE      23    /* OBJECT in TREE #0 */
#define SKIP        16    /* OBJECT in TREE #0 */
#define LMAR        33    /* OBJECT in TREE #0 */
#define RMAR        42    /* OBJECT in TREE #0 */
#define SET         20    /* OBJECT in TREE #0 */
#define PASET       19    /* OBJECT in TREE #0 */
#define PAPAGE      8     /* OBJECT in TREE #0 */
#define PASKIP      15    /* OBJECT in TREE #0 */
#define PALSPACE    22    /* OBJECT in TREE #0 */
#define PALMAR      32    /* OBJECT in TREE #0 */
#define PARMAR      41    /* OBJECT in TREE #0 */
/*-----------------------------------------------------------------------*/
/*      Fehlermeldung                                                     */
/*-----------------------------------------------------------------------*/
BYTE fehler[] = {
"[3][     !!! FEHLER !!!|Es ist kein Drucker ange-|schlossen oder der \
Drucker|ist nicht empfangsbereit.][OK|ABBRUCH]" };
/*-----------------------------------------------------------------------*/
/*      fix_objects                                                       */
/*-----------------------------------------------------------------------*/
VOID fix_objects()                         /* Setzen der Objektstrukturadressen*/
{                                          /*                                 */
 WORD   test, ii;                          /* Objektindex, Zaehler            */
 for(ii = 0; ii < NUM_OBS; ii++) {         /* Schleife ueber Anzahl Objekte   */
   test = (WORD) rs_object[ii].ob_spec; /*                                */
   rs_object[ii].ob_x      *= bchar;    /* x-Koord. * Zeichenbreite (Pixel) */
   rs_object[ii].ob_y      *= hchar;    /* y-Koord. * Zeichenhoehe  (Pixel) */
   if(rs_object[ii].ob_height == 1536) { /* wenn kleine Schrift            */
     rs_object[ii].ob_width  = rs_tedinfo[test].te_txtlen - 1; /* Laenge  */
     rs_object[ii].ob_height = 1;       /* Hoehe                          */
   }                                    /*                                 */
   rs_object[ii].ob_width  *= bchar;    /* Breite * Zeichenbreite (Pixel)  */
   rs_object[ii].ob_height *= hchar;    /* Hoehe  * Zeichenbreite (Pixel)  */
   switch (rs_object[ii].ob_type) {     /* Zeigeradressen entsprechend des */
          case G_TITLE:                 /* Objekt-Typs setzen              */
          case G_STRING:                /*                                 */
          case G_BUTTON:                /*                                 */
               fix_str(&rs_object[ii].ob_spec); /* Zeiger auf Text-String  */
               break;                   /*                                 */
          case G_TEXT:                  /*                                 */
          case G_BOXTEXT:               /*                                 */
          case G_FTEXT:                 /*                                 */
          case G_FBOXTEXT:              /*                                 */
               if(test != NIL)          /* wenn nicht (-1), Zeiger auf die */
                 rs_object[ii].ob_spec = (LONG) (&rs_tedinfo[test]); /*    */
               break;                   /* TEDINFO-Adresse setzen          */
          default:                      /*                                 */
   }                                    /* Ende 'case'-Anweisung           */
 }                                      /* Ende 'for'-Schleife             */
}                                       /* Ende der Funktion               */
/*-----------------------------------------------------------------------*/
/*      fix_tedinfo                                                       */
/*-----------------------------------------------------------------------*/
VOID fix_tedinfo()                      /* TEDINFO-Adressen setzen         */
{                                       /*                                 */
 WORD    ii;                            /* Zaehler                         */
 for(ii = 0; ii < NUM_TI; ii++) {       /* Schleife ueber Anzahl TEDINFO's */
   fix_str(&rs_tedinfo[ii].te_ptext);   /* Adresse fuer te_ptext           */
   fix_str(&rs_tedinfo[ii].te_ptmplt);  /* Adresse fuer te_ptmplt          */
   fix_str(&rs_tedinfo[ii].te_pvalid);  /* Adresse fuer te_pvalid          */
 }                                      /* Ende der Schleife               */
}                                       /* Ende der Funktion               */
/*-----------------------------------------------------------------------*/
```

```
/*     fix_str                                                          */
/*--------------------------------------------------------------------*/
VOID fix_str(where)                   /* Adresse des Stringanfangs wird  */
LONG   *where;                        /* an die in 'where' angegebene    */
{                                     /* Speicherstelle geschrieben      */
 if (*where != NIL)                   /* wenn ungleich (-1)              */
   *where=(LONG)(rs_strings[(WORD) *where]); /* LONG-Adresse             */
}                                     /* Ende der Funktion               */
/*--------------------------------------------------------------------*/
/*     do_obj                                                           */
/*--------------------------------------------------------------------*/
VOID do_obj(which, bit)               /* Objektbehandl. d. Objektbaumes  */
WORD   which, bit;                    /* welches Objekt, Bit             */
{                                     /*                                 */
 WORD state;                          /* Status                          */
 state = LWGET(OB_STATE(which));      /* Status des Objekts holen        */
 LWSET(OB_STATE(which), state | bit );  /* Bit im Objektstatus setzen    */
}                                     /* Ende der Funktion               */
/*--------------------------------------------------------------------*/
/*     undo_obj                                                         */
/*--------------------------------------------------------------------*/
VOID undo_obj(which, bit)             /* Objektbehandl. d. Objektbaumes  */
WORD which, bit;                      /* welches Objekt, Bit             */
{                                     /*                                 */
 WORD state;                          /* Status                          */
 state = LWGET(OB_STATE(which));      /* Status des Objekts holen        */
 LWSET(OB_STATE(which), state & ~bit ); /* Bit im Objektstatus loeschen  */
}                                     /* Ende der Funktion               */
/*--------------------------------------------------------------------*/
/*     select_obj                                                       */
/*--------------------------------------------------------------------*/
VOID select_obj(which)                /* Objekt als angewaehlt darstellen */
WORD   which;                         /* welches Objekt                  */
{                                     /*                                 */
 do_obj(which,SELECTED);              /* Objekt anwaehlen                */
}                                     /* Ende der Funktion               */
/*--------------------------------------------------------------------*/
/*     deselect_obj                                                     */
/*--------------------------------------------------------------------*/
VOID deselect_obj(which)              /* Obj. als nicht angewaehlt darst. */
WORD   which;                         /* welches Objekt                  */
{                                     /*                                 */
 undo_obj(which,SELECTED);            /* Objekt nicht anwaehlen          */
}                                     /* Ende der Funktion               */
/*--------------------------------------------------------------------*/
/*      get_parent                                                      */
/*--------------------------------------------------------------------*/
WORD get_parent(obj)                  /* findet Wurzel des Objekts       */
WORD   obj;                           /* Objekt                          */
{                                     /*                                 */
 WORD   pobj;                         /* Eltern-Objekt                   */
 if(obj == NIL)                       /* wenn Objekt = Wurzelobjekt      */
   return (NIL);                      /* Ruecksprung mit NIL             */
 pobj = LWGET(OB_NEXT(obj));          /* naechstes Objekt holen          */
 if(pobj != NIL) {                    /* wenn kein Wurzelobjekt          */
   while( LWGET(OB_TAIL(pobj)) != obj ) {  /* suche bis letztes Objekt   */
        obj = pobj;                   /* vertauschen                     */
        pobj = LWGET(OB_NEXT(obj));   /* naechstes Objekt                */
   }                                  /*                                 */
 }                                    /*                                 */
 return(pobj);                        /* Ruecksprung mit Eltern-Objekt   */
}                                     /* Ende der Funktion               */
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Fortsetzung)

```
/*-----------------------------------------------------------------------*/
/*      objc_xywh                                                        */
/*-----------------------------------------------------------------------*/
VOID objc_xywh(obj, p)                     /* holt x,y,w,h des Objektes   */
WORD obj;                                  /* Objekt                      */
GRECT *p;                                  /* Zeiger auf Struktur fuer x,y,w,h */
{                                          /*                             */
 objc_offset(tree, obj, &p->g_x, &p->g_y); /* x,y-Koordinaten des Objekts */
 p->g_w = LWGET(OB_WIDTH(obj));            /* w des Objekts               */
 p->g_h = LWGET(OB_HEIGHT(obj));           /* h des Objekts               */
}                                          /* Ende der Funktion           */
/*-----------------------------------------------------------------------*/
/*      do_string                                                        */
/*-----------------------------------------------------------------------*/
VOID do_string(obj, num)                   /* Zahl in Objekt-String schreiben */
WORD obj;                                  /* Objekt                      */
WORD num;                                  /* Zahl                        */
{                                          /*                             */
 BYTE   *ptr;                              /* Zeiger auf String           */
 WORD   length;                            /* Stringlaenge                */
 ptr = (BYTE *) LLGET(LLGET(OB_SPEC(obj)));  /* Zeiger ermitteln          */
 length = strlen(ptr);                     /* Laenge des Strings          */
 ptr += length;                            /* Stringpointer auf letztes Zeichen*/
 do{                                       /* Schleife ueber Anzahl der Zahlen */
  *--ptr = num % 10 + '0';                 /* MOD(num,10)=letzte Stelle->String*/
  length--;                                /* Zeichenzaehler - 1          */
 } while((num /= 10) > 0);                 /* num=INT(num/10),Schleife b. num=0*/
 while(length-- != 0)                      /* String mit Leerzeichen auffuellen*/
  *--ptr = ' ';                            /* bis Zeichenzaehler = 0 ist  */
}                                          /* Ende der Funktion           */
/*-----------------------------------------------------------------------*/
/*      prout                                                            */
/*-----------------------------------------------------------------------*/
VOID prout()                               /* Abfrage d. Tasten u. Druckerausg.*/
{                                          /*                             */
 WORD i;                                   /* Zaehler                     */
 if(LWGET(OB_STATE(RESON)) & SELECTED)     /* Taste RESET ON ?            */
    output(p_reset);                       /*                             */
 if(LWGET(OB_STATE(UNIDIR)) & SELECTED)    /* Taste UNIDIR ?              */
    output(p_unidir);                      /*                             */
  else                                     /* Nein: also BIDIR            */
    output(p_bidir);                       /*                             */
 output(p_set);                            /* Zeichensatz                 */
 output(p_page);                           /* Seitenlaenge                */
 if(p_skip[2] == 0)                        /* wenn kein Perforations-Sprung */
    output(p_offskip);                     /* Sprung loeschen             */
  else                                     /* sonst                       */
    output(p_skip);                        /* Sprung setzen               */
 output(p_lmar);                           /* linker Druckrand            */
 output(p_rmar);                           /* rechter Druckrand           */
 output(p_lspace);                         /* Zeilenabstand               */
 if(LWGET(OB_STATE(MICROON)) & SELECTED) /* Taste MICRO ON ?              */
    output(p_micro);                       /*                             */
  else                                     /* Nein: also MICRO OFF        */
   { if(LWGET(OB_STATE(ELITEON)) & SELECTED) /* Taste ELITE ?             */
       output(p_elite);                    /*                             */
     else                                  /* Nein: weiter abfragen       */
      { if(LWGET(OB_STATE(CONDON)) & SELECTED) /* Taste COND ?            */
         output(p_cond);                   /*                             */
        else                               /* Nein: also PICA             */
         output(p_pica);                   /*                             */
      }                                    /*                             */
   }                                       /*                             */
```

```
 if(LWGET(OB_STATE(NLQON)) & SELECTED)  /* Taste NLQ ?                              */
   { output(p_pica);                    /*                                         */
     output(p_nlq);  }                  /*                                         */
   else                                 /* Nein: weiter abfragen                   */
   { if(LWGET(OB_STATE(ITALON)) & SELECTED) /* Taste ITALIC ?                      */
         output(p_ital);                /*                                         */
       else                             /* Nein: also STANDARD                     */
         output(p_stand);               /*                                         */
   }                                    /*                                         */
 if(LWGET(OB_STATE(EXPON)) & SELECTED)  /* Taste EXPANDED ON ?                     */
     output(p_onexp);                   /*                                         */
   else                                 /* Nein: also EXPANDED OFF                 */
     output(p_offexp);                  /*                                         */
 if(LWGET(OB_STATE(PROPON)) & SELECTED) /* Taste PROPORT ON ?                      */
     output(p_onprop);                  /*                                         */
   else                                 /* Nein: also PROPORT OFF                  */
     output(p_offprop);                 /*                                         */
 if(LWGET(OB_STATE(DOUBON)) & SELECTED) /* Taste DOUBLE ON ?                       */
     output(p_ondoub);                  /*                                         */
   else                                 /* Nein: also DOUBLE OFF                   */
     output(p_offdoub);                 /*                                         */
 if(LWGET(OB_STATE(EMPHON)) & SELECTED) /* Taste EMPHASIZED ON ?                   */
     output(p_onemph);                  /*                                         */
   else                                 /* Nein: also EMPHASIZED OFF               */
     output(p_offemph);                 /*                                         */
 if(LWGET(OB_STATE(ZEROON)) & SELECTED) /* Taste durchstrichene Null ?             */
     output(p_onzero);                  /*                                         */
   else                                 /* Nein: also normale Null                 */
     output(p_offzero);                 /*                                         */
 output(p_ret);                         /* Return ausgeben                         */
 if(LWGET(OB_STATE(PRINTON)) & SELECTED)  /* Test-Ausdruck ?                       */
   { Cprnout(10);                       /* Linefeed                                */
     for(i=32;i<127;i++)                /* ASCII-Satz                              */
        Cprnout(i);                     /* Zeichen ausgeben                        */
     Cprnout(10);                       /* Linefeed                                */
   }                                    /*                                         */
 if(LWGET(OB_STATE(FORMON)) & SELECTED) /* Taste Seitenvorschub ?                  */
     output(p_onform);                  /*                                         */
}                                       /* Ende der Funktion                       */
/*-----------------------------------------------------------------------------*/
/*        output                                                               */
/*-----------------------------------------------------------------------------*/
VOID output(pointer)                    /* Ausgabe Control-Codes bis $FF           */
WORD  *pointer;                         /* Zeiger auf Ausgabebereich               */
{                                       /*                                         */
 while( *pointer != 255 )               /* solange Zeichen ausgeben bis $FF        */
      Cprnout(*pointer++);              /* ein Zeichen an Ausgabegeraet            */
}                                       /* Ende der Funktion                       */
/*-----------------------------------------------------------------------------*/
/*        do_dialog                                                            */
/*-----------------------------------------------------------------------------*/
VOID do_dialog()                        /* Dialogbehandlung                        */
{                                       /*                                         */
 LONG  printer = NO;                    /* Druckeranwesenheitsflag                 */
 WORD  xdial,ydial,bdial,hdial;         /* Koordinaten                             */
 WORD  parent,exit_obj;                 /* Eltern- und Exit-Objekt                 */
 WORD  xtype;                           /* Exit-Pfeil                              */
 GRECT ob;                              /* Struktur fuer x,y,w,h exit_obj          */
 form_center(tree,&xdial,&ydial,&bdial,&hdial); /* Koordinaten holen               */
 form_dial(0,1,1,1,1,xdial,ydial,bdial,hdial);  /* Bereich sichern                 */
 form_dial(1,1,1,1,1,xdial,ydial,bdial,hdial);  /* aufgehender Kasten              */
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Fortsetzung)

```
objc_draw(tree,0,10,xdial,ydial,bdial,hdial); /* Baumobjekt zeichnen         */
                                             /*                             */
FOREVER                                      /* Endlos-Schleife             */
{                                            /*                             */
  exit_obj = form_do(tree, 0) & 0x7FFF; /* Objekt-Kontrolle abgeben         */
  xtype = LWGET(OB_TYPE(exit_obj)) & 0xFF00; /* Exit-Pfeil herausshiften    */
  if((!xtype) && (exit_obj != SET))          /* wenn kein Exitpfeil od. SET-Feld */
    break;                                   /* Verlassen der Endlos-Schleife */
                                             /*                             */
  xtype = (xtype == X_BAK) ? -1 : 1;         /* xtype auf +1 oder -1 setzen */
  parent = get_parent(exit_obj);             /* Objekt-Index des Eltern-Objekts */
        switch(parent) {                     /* Funktionen je nach Eltern-Objekt */
          case PASET:                        /* wenn Zeichensatz            */
               exit_obj = SET;               /* neu zu zeichnendes Objekt   */
               p_set[2] = (p_set[2] == MAX_SET) ? MIN_SET : ++p_set[2];
               LLSET(OB_SPEC(SET),(LONG)str_set[p_set[2]]); /* String-      */
               break;                        /* pointer erhoehen, umsetzen  */
          case PAPAGE:                       /* wenn Seitenlaenge           */
               exit_obj = PAGE;              /* neu zu zeichnendes Objekt   */
               p_page[3] = (p_page[3]+xtype) % (MAX_PAGE+1);
               if (p_page[3] < MIN_PAGE)
                  p_page[3] = (xtype == -1) ? MAX_PAGE : MIN_PAGE;
               do_string(PAGE,p_page[3]);
               break;                        /*                             */
          case PASKIP:                       /* wenn Seitenperforation      */
               exit_obj = SKIP;              /* neu zu zeichnendes Objekt   */
               p_skip[2] = (p_skip[2]+xtype) % (MAX_SKIP+1);
               if (p_skip[2] < MIN_SKIP)
                  p_skip[2] = (xtype == -1) ? MAX_SKIP : MIN_SKIP;
               do_string(SKIP,p_skip[2]);
               break;                        /*                             */
          case PALSPACE:                     /* wenn Zeilenabstand          */
               exit_obj = LSPACE;            /* neu zu zeichnendes Objekt   */
               p_lspace[2] = (p_lspace[2]+xtype) % (MAX_LSPA+1);
               if (p_lspace[2] < MIN_LSPA)
                  p_lspace[2] = (xtype == -1) ? MAX_LSPA : MIN_LSPA;
               do_string(LSPACE,p_lspace[2]);
               break;                        /*                             */
          case PALMAR:                       /* wenn linker Druckrand       */
               exit_obj = LMAR;              /* neu zu zeichnendes Objekt   */
               p_lmar[2] = (p_lmar[2]+xtype) % p_rmar[2];
               if (p_lmar[2] < MIN_LMAR)
                  p_lmar[2] = (xtype == -1) ? (p_rmar[2]-1) : MIN_LMAR;
               do_string(LMAR,p_lmar[2]);
               break;                        /*                             */
          case PARMAR:                       /* wenn rechter Druckrand      */
               exit_obj = RMAR;              /* neu zu zeichnendes Objekt   */
               p_rmar[2] = (p_rmar[2]+xtype) % (MAX_RMAR+1);
               if (p_rmar[2] < (p_lmar[2]+1))
                  p_rmar[2] = (xtype == -1) ? MAX_RMAR : (p_lmar[2]+1);
               do_string(RMAR,p_rmar[2]);
          default: break;                    /*                             */
        }                                    /*                             */
  objc_xywh(exit_obj,&ob);                   /* x,y,w,h des Objekts holen   */
  objc_draw(tree,parent,1,ob.g_x+1,ob.g_y+1,ob.g_w-2,ob.g_h-2);
}                                            /* Ende der Endlos-Schleife    */
                                             /*                             */
if(exit_obj == OKAY) {                       /* wenn D.Ausgabe D.Status abfragen */
  while(printer == NO) {                     /* Schleife ueber Druckeranwesenheit*/
    printer = Cprnos();                      /* Druckerstatus abfragen      */
    if(printer == NO) {                      /* wenn kein Drucker           */
      if(form_alert(2,fehler) == 2) {        /* Ausgabe Fehlermeldung       */
        printer = YES;                       /* wenn Abbruch gefordert wurde */
```

```
        exit_obj = CANCEL;                    /* Abbruch der Ausgabe setzen und     */
      }                                       /* keine Druckerausgabe               */
    }                                         /*                                    */
  }                                           /* Ende der while-Schleife            */
 }                                            /* Ende der Anwesenheitsabfrage       */
 form_dial(2,1,1,1,1,xdial,ydial,bdial,hdial); /* zugehender Kasten                 */
 form_dial(3,1,1,1,1,xdial,ydial,bdial,hdial); /* Bereich herstellen                */
 if(exit_obj == OKAY) {                       /* Druckercodes abschicken            */
   prout();                                   /* Ausgabe der Drucker-Codes          */
 }                                            /*                                    */
 deselect_obj(OKAY);                          /* nicht angewaehlt darstellen        */
 deselect_obj(CANCEL);                        /* nicht angewaehlt darstellen        */
}                                             /* Ende der Funktion                  */
/*--------------------------------------------------------------------------------*/
/*        do_vwork                                                                 */
/*--------------------------------------------------------------------------------*/
VOID do_vwork()                               /* Initialisierung der Anwendung      */
{                                             /*                                    */
WORD i;                                       /* Zaehler                            */
 for(i=0; i<10; work_in[i++]=1);              /* workin-Array initialisieren        */
 work_in[10] = 2;                             /* Bildschirmkoordinaten              */
 v_opnvwk(work_in,&phys_handle,work_out); /* anmelden der Arbeitsstation            */
}                                             /* Ende der Funktion                  */
/*--------------------------------------------------------------------------------*/
/*        Event-Handler                                                            */
/*--------------------------------------------------------------------------------*/
VOID h_event()                                /*                                    */
{                                             /*                                    */
       FOREVER                                /* Endlosschleife                     */
       {                                      /*                                    */
        event = evnt_multi(MU_MESAG,          /* warten auf eine Nachricht          */
                 1,1,                         /* 1 Maustastendruck von Maustaste 1*/
                 1,                           /* fuer Taste unten                   */
                 0,0,0,0,0,                   /*                                    */
                 0,0,0,0,0,                   /*                                    */
                 msgbuff,                     /* Adresse Buffer fuer Nachrichten    */
                 &ret,&ret,                   /* nicht benutzte Funktionen          */
                 &ret,&ret,&ret,&ret);        /*                                    */
         if (event & MU_MESAG) {              /* Nachricht?                         */
           switch (msgbuff[0]) {              /* Menuewahl                          */
           case AC_OPEN:                      /* Desk-Accessory angewaehlt          */
               if (msgbuff[4] == menu_id) { /* eigenes Programm gewaehlt?           */
                 wind_update(BEG_UPDATE); /* keine Benutzeraktivitaet zulassen*/
                 do_vwork();                  /* Arbeit initialisieren              */
                 do_dialog();                 /* Dialogbehandlung                   */
                 v_clsvwk(phys_handle);       /* Arbeitsstation schliessen          */
                 wind_update(END_UPDATE); /* Benutzeraktivitaet zulassen            */
               }                              /*                                    */
           default: break;                    /*                                    */
           }                                  /* Ende der switch-Anweisung          */
         }                                    /* Ende der if-Anweisung              */
       }                                      /* Schleifenende                      */
}                                             /* Ende der Funktion                  */
/*********************************************************************************/
/****                          Hauptprogramm                                ****/
/*********************************************************************************/
main()                                        /*                                    */
{                                             /*                                    */
 appl_init();                                 /* initialisieren der Applikation     */
 phys_handle = graf_handle(&bchar,&hchar,&bbox,&hbox); /* handle holen             */
 fix_objects();                               /* OBJEKT-Struktur initialisieren     */
 fix_tedinfo();                               /* TEDINFO-Struktur initialisieren    */
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Fortsetzung)

```
 tree = (LONG)rs_object;                          /* Anfangsadresse der Baumstruktur  */
  menu_id = menu_register(gl_apid,MENU_NAME); /* Menue-Eintrag vornehmen          */
  select_obj(BIDIR);                              /* Bidirektionaler Druck            */
  select_obj(PICAON);                             /* Pica an                          */
  select_obj(EXPOFF);                             /* Expanded aus                     */
    do_obj(PROPON,DISABLED);                      /* Taste Proport. AN nicht waehlbar */
    do_obj(PROPOFF,DISABLED);                     /* Taste Proport. AUS nicht waehlbar*/
  select_obj(RESOFF);                             /* Drucker-Reset aus                */
  select_obj(FORMOFF);                            /* Seitenvorschub aus               */
  select_obj(STANDON);                            /* Standard an                      */
  select_obj(DOUBOFF);                            /* Doppeldruck aus                  */
  select_obj(EMPHOFF);                            /* Emphasizeddruck aus              */
    do_obj(ZEROON,DISABLED);                      /* Taste Null AN nicht waehlbar     */
    do_obj(ZEROOFF,DISABLED);                     /* Taste Null AUS nicht waehlbar    */
  select_obj(PRINTOFF);                           /* kein Testdruck                   */
    do_obj(NLQON,DISABLED);                       /* Taste NLQ nicht waehlbar         */
  LLSET(OB_SPEC(SET),(LONG)str_set[p_set[2]]); /* Zeichensatz                      */
  do_string(PAGE,p_page[3]);                      /* Seitenlaenge                     */
  do_string(SKIP,p_skip[2]);                      /* Perforations-Sprung              */
  do_string(LSPACE,p_lspace[2]);                  /* Zeilenabstand                    */
  do_string(LMAR,p_lmar[2]);                      /* linker Druckrand                 */
  do_string(RMAR,p_rmar[2]);                      /* rechter Druckrand                */
  if( Cprnos() != NO )                            /* wenn Drucker vorhanden           */
    prout();                                      /* Grundeinstellung initialisieren  */
  h_event();                                      /* Event-Handler (Endlosschleife)   */
}                                                 /* Ende main                        */
/**** ende *******************************************************************/
/*****************************************************************************/
```

Komfortables Accessory für jeden Epson-kompatiblen Drucker (Schluß)

Blitzfloppy mit ROM-ST

Die gewohnten Schnelladeprogramme für den Atari ST laufen leider nicht unter der ROM-Version des TOS. »FLOAD V2 ST« überlistet das ROM-TOS und macht dem Laufwerk wieder Beine.

Wen wurmt das nicht: Da hat man ein Programm in seiner Sammlung, das beim Diskettenbetrieb ordentlich Dampf macht. Leider muß man jedoch für dieses Programm das Betriebssystem TOS von der Diskette booten, statt die teuren neuen ROMs zu nutzen. Gängige Programme wie zum Beispiel »Fastload ST« (siehe Happy-Computer 5/86) verändern nämlich bestimmte Speicherstellen direkt im Betriebssystem. Man nennt diesen Vorgang im Computerchinesisch »Patchen«. Patchen kann man aber nur Speicherstellen im Schreib-Lese-Speicher des Computers (RAM). Leider befinden sich die entscheidenden Speicherplätze in ST-Computern mit ROM-Betriebssystem – wie sollte es auch anders sein – im Festspeicherbereich des ST. Wie schon der kleine Computer-Moritz weiß, lassen sich ROM-Speicherplätze selbstverständlich nicht beschreiben. Deshalb aber auf eine schnelle Floppy verzichten? Nein, der Einfallsreichtum der Programmiererzunft läßt sich durch solche Widrigkeiten nicht abschrecken! Das Ergebnis solchen Programmiererfleißes sehen Sie in unseren DATA-Zeilen vor sich.

»FLOAD V2 ST« basiert auf der gleichen Grundidee wie schon das bewährte »Fastload ST«. Das Betriebssystem TOS fragt nämlich nach jeder Positionierung des Schreib/Lese-Kopfes die gesetzte Position zur Kontrolle noch einmal ab. Bei guten Diskettenlaufwerken erübrigen sich eigentlich solche Vorsichtsmaßnahmen. Durch Abschalten der Kontrollabfrage läßt sich viel Zeit gewinnen. Im Unterschied zu »Fastload ST« wird zunächst im RAM eine neue Diskettenzugriffsroutine aufgebaut. »FLOAD V2 ST« kopiert die Originalroutine (etwa 10 KByte) aus dem Betriebssystem-ROM in den RAM-Bereich und reloziert sie (paßt sie also an die neue RAM-Adresse an). Anschließend werden die Änderungen vorgenommen, die die Kontrollabfragen abschalten. Zu guter Letzt teilt man dem Betriebssystem noch mit, wo die neue Routine zu finden ist. Dazu wird der Zeiger auf die Diskettenroutine (Speicherstellen $476, $477, $478 und $479), der in den ROM-Bereich weist, auf die Adresse der veränderten Routine im RAM umgestellt.

Um böse Überraschungen (Systemabsturz oder ähnliches) zu vermeiden, prüft »FLOAD V2 ST« vor der Installation der Schnelladeroutine, ob das TOS im ROM-Bereich präsent ist. Bei negativem Ergebnis wird untersucht, welche Betriebssystemversion im RAM-Bereich vorliegt. Findet das Programm eine TOS-Version älteren Datums als November 1985 vor, wird »FLOAD V2 ST« nicht gestartet und statt dessen eine entsprechende Meldung ausgegeben.

Wie auch schon mit »Fastload ST« gibt es Einschränkungen für die Arbeit mit »FLOAD V2 ST«. Beim Disketten-Backup aus dem Desktop kann es zu Schwierigkeiten kommen, da die Bildschirmgrafik der Geschwindigkeit der Diskettenlaufwerke nicht gewachsen ist. Andere Kopierprogramme sollten

allerdings ohne Probleme laufen. Der Schnellformatierer wurde nicht installiert.

Der vorliegende Basic-Lader erzeugt auf einer Diskette das Programm »FLOAD V2.PRG«, das durch Doppelklick vom Desktop aus gestartet werden kann. Automatisches Starten läßt sich durch Speichern in einem Auto-Ordner auf der Startdiskette erreichen. Man sollte darauf achten, daß »FLOAD V2.PRG« sich als erstes Programm im Ordner befindet (Speichern in einen leeren Ordner). Nur in diesem Fall gelangt man so früh wie möglich in den Genuß der Schnelladerei. Schon das nächste Programm im Auto-Ordner oder die Desktop-Accessories auf der Startdiskette werden schneller geladen. Der Betrieb einer Festplattenstation stößt auch mit Schnellader nicht auf Bedenken.

Haben Sie unser Listing richtig abgetippt, so war das ST-Basic eines der letzten Programme, das Ihnen wertvolle Zeit beim Diskettenbetrieb gestohlen hat.

(M. Bernards/W. Fastenrath/hb)

```
115   goto start
120   add:z=0
130   z=z+1
140   for i=1 to 70
150   read a:if a<0 then return
160   a(z)=a(z)+a
170   next i
180   goto 130
190   return
200   pruef:
210   for i=1 to z
220   read a
230   if a<>a(i) then goto fehler
240   next i
250   return
490   prggen:
500   open "R",#1,prgnam$,2
510   field #1, 2 as a$
520   i=0
530   i=i+1
540   read b:if b<0 then 590
550   read c:if c<0 then 590
560   d=256*b+c:lset a$=mki$(d)
570   put #1,i
580   goto 530
590   close:return
790   fehler:
800   fullw 2:clearw 2:gotoxy 0,0
810   print "FEHLER ZWISCHEN DATAZEILE";
820   print zeile + (i-1) * 100;" UND";
830   print zeile + i * 100
840   end
890   rem ***** PRUEFSUMMEN *****
900   data 3090,3851,4283,5643,6667,4502
910   data 2882,-1
990   rem ***** PROGRAMMDATAS *****
1000  data 096,026,000,000,001,182,000
1010  data 000,000,000,000,000,000,000
1020  data 000,000,000,000,000,000,000
1030  data 000,000,000,000,000,000,000
1040  data 042,111,000,004,044,045,000
1050  data 012,220,173,000,020,220,173
1060  data 000,028,006,134,000,000,032
1070  data 000,035,198,000,000,000,122
1080  data 012,185,174,214,140,023,000
1090  data 252,029,008,103,000,000,026
1100  data 012,121,124,020,000,000,122
1110  data 029,102,000,000,074,019,252
1120  data 000,016,000,000,122,029,096
1130  data 000,000,044,072,121,000,000
1140  data 001,038,063,060,000,038,078
1150  data 078,097,000,000,240,072,121
1160  data 000,000,001,138,063,060,000
1170  data 038,078,078,072,121,000,000
1180  data 000,158,063,060,000,009,078
1190  data 065,092,143,066,103,047,057
1200  data 000,000,000,122,063,060,000
1210  data 049,078,065,000,000,000,049
1220  data 019,252,000,013,000,000,000
1230  data 232,072,121,000,000,000,158
1240  data 063,060,000,009,078,065,092
1250  data 143,063,060,000,007,078,065
1260  data 066,087,078,065,027,069,032
1270  data 010,027,112,070,097,115,116
1280  data 108,111,097,100,101,114,032
1290  data 118,050,046,048,032,098,121
1300  data 032,077,105,099,104,097,101
1310  data 108,032,066,101,114,110,097
1320  data 114,100,115,032,102,129,114
1330  data 032,072,097,112,112,121,032
1340  data 067,111,109,112,117,116,101
1350  data 114,032,117,110,100,032,054
1360  data 056,048,048,048,101,114,027
1370  data 113,000,010,010,032,070,097
1380  data 108,115,099,104,101,115,032
1390  data 084,079,083,032,033,033,032
1400  data 070,097,115,116,108,111,097
1410  data 100,101,114,032,107,097,110
1420  data 110,032,110,105,099,104,116
1430  data 032,105,110,115,116,097,108
1440  data 108,105,101,114,116,032,119
1450  data 101,114,100,101,110,000,000
1460  data 065,249,000,252,000,000,067
1470  data 249,000,000,001,182,032,060
1480  data 000,000,008,000,034,216,081
1490  data 200,255,252,078,117,065,249
1500  data 000,000,001,118,074,144,103
1510  data 000,000,026,034,088,211,252
1520  data 000,000,001,182,004,145,000
1530  data 252,000,000,006,145,000,000
1540  data 001,182,096,000,255,228,032
1550  data 124,000,000,027,143,209,252
1560  data 000,000,001,182,016,188,000
1570  data 016,078,117,000,000,018,050
1580  data 000,000,018,096,000,000,018
1590  data 150,000,000,019,200,000,000
1600  data 000,000,035,249,000,000,004
1610  data 118,000,000,001,178,035,252
1620  data 000,000,001,160,000,000,004
1630  data 118,078,117,012,111,000,002
1640  data 000,014,107,000,016,224,032
1650  data 121,000,000,001,178,078,208
1660  data 000,000,000,000,000,000,000
1670  data 024,044,016,012,016,018,006
1680  data 166,020,014,012,016,036,006
1690  data 022,000,-1
4000  start:
4010  clear:restore 1000:dim a(30)
4015  for i=0 to 30:a(i)=0:next
4020  gosub add
4030  restore 900
4040  zeile=1000:gosub pruef
4050  prgnam$ = "FLOAD_V2.PRG"
4060  restore 1000:gosub prggen
4070  end
```

Listing. »FLOAD V2« bringt Ihre Disketten auf Touren

ST-Menü à la Carte

Welches Menü bietet der Atari ST? Das fragen sich in Zukunft auch viele Basic-Programmierer. »GEM-Menü« macht Maussteuerung und Pull-down-Menüs kinderleicht.

Pull-down-Menüs und Maussteuerung in ST-Basic selbst programmieren ist schwierig. Im Sprachumfang fehlen Kommandos, um GEM auf einfache Weise anzusprechen. Man muß auf POKE-Befehle zurückgreifen. Damit klarzukommen, dauert. Noch dazu schweigt sich das Basic-Handbuch leider über die Anwendung der Befehle aus. Gerade bei einem Computer der »neuen« Generation enttäuscht ein Basic-Interpreter, der noch der »alten« Generation angehört.

Um Sie vor der Plage von POKE-Befehlen zu bewahren, stellen wir Ihnen ein Allround-Menü vor. Sobald das Listing eingetippt und mit Run gestartet ist, erscheinen auf dem Bildschirm vier Titel. Die Bedienung ist kinderleicht. Um das jeweilige Pull-down-Menü unter einem Titel herunterzuklappen, bewegen Sie den Mauszeiger auf den betreffenden Titel und drücken eine Maustaste. Sofort erscheint das Menü. Zum Teil füllen Platzhalter die Zeilen, andere enthalten bereits bestimmte Funktionen.

Das erste Menü zeigt das Atari-Firmenlogo. Klicken Sie es an, öffnet sich ein Kommentarfenster auf dem Bildschirm. Darin können Sie zum Beispiel Informationen über das Programm und über den Autor verewigen. Nach einem Klick auf den »M M-Knopf« verschwindet das Fenster wieder vom Bildschirm.

Das Pull-down-Menü unter dem zweiten Titel mit Namen »Info« zeigt nach dem Anklicken den Text »About GEM__Menu«. Klicken Sie darauf, füllt ein Arbeitsfenster den ganzen Bildschirm aus. Es demonstriert die Nutzung eines Arbeitsfensters. Es empfiehlt sich, Eingaben durch »INP (2)« vorzunehmen. Um »INPUT« zu verwenden, muß man vorher den Befehl »POKE SYSTAB+24,0« ausführen. Das aktiviert allerdings auch das Basic-Desktop wieder und kann »GEM-Menü« stören.

Um das Programm zu beenden und zum Desktop zurückzukehren, konsultieren Sie das dritte Pull-down-Menü. Die unterste Funktion »Quit« gibt ein kleines Fenster frei, in dem Sie durch einen Mausklick bestätigen, daß Sie das Programm wirklichen beenden möchten.

Sie dürfen anstelle der Platzhalter auch beliebige andere Texte einsetzen. Diese beeinflussen das Programm in keiner Weise.

Die Funktionen »Unterprogramm 1 bis 9« und »Option 1 bis 3« sind durch REM-Befehl stillgelegt. An dieser Stelle baut man eigene Routinen ein.

Wer nun neugierig auf den Aufbau und die Programmierung solcher komfortabler Menüs geworden ist, dem liefert dieses Beispiel eine ideale Vorlage.

(Marco Meyer/hb)

```
10 # GEM-MENÜ für das Einbinden von Basic-Programmen          #
20 # MARCO MEYER  GERHARD-ROHLFS-STRASSE 54c  D-282 BREMEN 70 #
30 # Outputfenster auf maximale Groesse, loeschen und Mauszeiger aus! #
40 openw 2,0,1,660,420:clearw 2:gosub maus
50 goto 900
60 # Simulation eines GEM-Fensters als Unterroutine zum Anspringen! #
70 fel:
80 xp1=-1:yp1=-19:xp2=634:yp2=361:gosub re:xp1=-1:yp1=-19:xp2=17:yp2=-1
90 gosub re:color 1,1,1,2,2:xp1=17:yp1=-19:xp2=616:yp2=-1:gosub re
100 color 1,1,1,0,0:xp1=615:yp1=-19:xp2=633:yp2=-1:gosub re
110 xp1=615:yp1=-1:xp2=633:yp2=16:gosub re:xp1=615:yp1=16
120 xp2=633:yp2=326:gosub re:xp1=615:yp1=326:xp2=633:yp2=344:gosub re
130 xp1=615:yp1=344:xp2=633:yp2=361:gosub re:xp1=-1:yp1=344:xp2=17:yp2=361
140 gosub re:xp1=17:yp1=344:xp2=597:yp2=361:gosub re:xp1=597:yp1=344:xp2=615
150 yp2=361:gosub re:m=2:gosub graf:xp=5:yp=-4:t$=chr$(5):gosub ta:xp=620
160 yp=-4:t$=chr$(240):gosub ta:xp=621:yp=359:t$=chr$(240):gosub ta:xp=620
170 yp=13:t$=chr$(1):gosub ta:xp=620:yp=341:t$=chr$(2):gosub ta:xp=5:yp=359
180 t$=chr$(4):gosub ta:xp=603:yp=359:t$=chr$(3):gosub ta:m=1:gosub graf
190 color 1,1,1,1,2:xp1=615:yp1=280:xp2=633:yp2=326:gosub re:xp1=570:yp1=344
200 xp2=597:yp2=361:gosub re:return
210 #    VDI-Routine fuer verschiedene Grafikmodi #
220 graf:
230 poke contrl,32
240 poke contrl+2,0
250 poke contrl+6,1
260 poke intin,m
270 vdisys
280 return
290 #VDI-Routine zur Ausgabe von Text nach Pos. und Masstab! #
300 ta:
310 for v=0 to len(t$)-1
320 poke intin+v*2,asc(mid$(t$,v+1,1))
330 next v
340 poke intin+v*2,0
350 poke contrl,8
360 poke contrl+2,1
370 poke contrl+6,len(t$)+1
380 poke ptsin,xp+1
390 poke ptsin+2,yp+38
400 vdisys
410 return
420 # VDI-Routine zum Einschalten des Maussymbols! #
430 mein:
440 poke contrl,122:poke intin,0:vdisys
450 return
460 # VDI-Routine zum Abschalten des Maussymbols! #
470 maus:
480 poke contrl,123:poke intin,0:vdisys
490 return
500 # VDI-Routine zum Bilden eines Rechteckes! #
510 re:
520 poke contrl,11
530 poke contrl+2,2
540 poke contrl+6,0
550 poke contrl+10,1
560 poke ptsin,xp1+1
570 poke ptsin+2,yp1+38
580 poke ptsin+4,xp2+1
590 poke ptsin+6,yp2+38
600 vdisys
610 return
620 # VDI-Routine zur Mausabfrage und Abschalten des Basicsystems! #
630 ze:
640 poke systab+24,1
650 gosub mein
660 poke contrl,124
670 vdisys
680 x=peek(ptsout)-1
690 y=peek(ptsout+2)-38
700 t=peek(intout)
710 if t=0 then goto ze
720 gosub maus
730 return
740 # Unterroutine zum Loeschen des Bildschirmes! #
750 hi:
760 color 1,1,1,4,2:xp1=-1:yp1=-20:xp2=638:yp2=362:gosub re:color 1,1,1,0,0
770 return
780 # Hilfsroutine fuer ein Rechteck #
790 re2:
800 gosub hi:xp1=137:yp1=47:xp2=503:yp2=303:gosub re:xp1=140:yp1=50:xp2=500
810 yp2=300:gosub re:xp1=141:yp1=51:xp2=499:yp2=299:gosub re:return
820 # Unterroutine zum Bilden einer Dialogbox in GEM Manier! #
830 rel:
840 xp1=166:yp1=96:xp2=474:yp2=204:gosub re:xp1=169:yp1=99:xp2=471
850 yp2=201:gosub re:xp1=170:yp1=100:xp2=470:yp2=200:gosub re:gotoxy 13,8
860 ?"Are you shure to execute?":xp1=219:yp1=164:xp2=281:yp2=183:gosub re
870 xp1=220:yp1=165:xp2=280:yp2=182:gosub re:xp1=357:yp1=164:xp2=419:yp2=183
880 gosub re:xp1=358:yp1=165:xp2=418:yp2=182:gosub re:xp=243:yp=179:t$="NO"
890 gosub ta:xp=378:t$="YES":gosub ta:return
900 gosub hi:xp1=-1:yp1=-40:xp2=638:yp2=-20:gosub re ##### START #####
910 # Bilden der Menueleiste! #
920 xp=20:yp=-23:t$=chr$(14)+chr$(15):gosub ta
930 xp=60:yp=-23:t$="INFO":gosub ta:xp=120:yp=-23:t$="MENÜ":gosub ta
940 xp=180:yp=-23:t$="SHOW":gosub ta:xp=240:yp=-23:t$="OPTIONS":gosub ta
```

```
950 # Hauptwarteroutine mit staendiger Abfrage von Maus und Menueleiste! #
960 gosub ze
970 if x>1 and  x<48 and y<-25 then goto atari
980 if x>=48 and x<106 and y<-25 then goto info
990 if x>=106 and x<166 and y<-25 then goto menu
1000 if x>=166 and x<227 and y<-25 then goto show
1010 if x>=227 and x<302 and y<-25 then goto options
1020 gosub hi
1030 goto 960
1040 # Erste freibelegbare und erweiterbare Hauptunterroutine! #
1050 atari:
1060 gosub hi:xp=9:yp=-2:t$=" Intersoft"+chr$(190):gosub ta:gosub ze
1070 if x<9 or x>104 or y>0 or y<-15 then goto 960
1080 gosub re2:xp=170:yp=80
1090 t$="INTERSOFT-THE UNCANNY ATARI COOPARATOR":gosub ta:xp=280:yp=123
1100 t$="INTERSOFT"+chr$(190):gosub ta:xp=240:yp=140:t$="-------------------"
1110 gosub ta:xp=295:yp=157:t$="ATARI"+chr$(191):gosub ta:xp=245:yp=190
1120 t$="Copyright (c) 1986":gosub ta:xp1=259:yp1=249:xp2=381:yp2=271:gosub re
1130 xp1=260:yp1=250:xp2=380:yp2=270:gosub re:xp=312:yp=266:t$="MM":gosub ta
1140 goto 960
1150 # Zweite freibelegbare und erweiterbare Hauptunterroutine! #
1160 info:
1170 gosub hi:xp=48:yp=-2:t$=" About GEM_MENU"+chr$(190):gosub ta:gosub ze
1180 if x<48 or x>159 or y<-15 or y>0 then 960
1190 gosub fe1:xp=220:yp=-5:t$="SAMPLE WINDOW TO GEM_MENU":gosub ta
1200 gosub ze
1210 goto 960
1220 # Dritte freibelegbare und erweiterbare Hauptunterroutine! #
1230 menu:
1240 gosub hi:xp=106
1250 yp=-2:t$=" UNTERPRG 001":gosub ta
1260 yp=17:t$=" UNTERPRG 002":gosub ta
1270 yp=36:t$=" UNTERPRG 003":gosub ta
1280 yp=55:t$=" UNTERPRG 004":gosub ta
1290 yp=74:t$=" UNTERPRG 005":gosub ta
1300 yp=93:t$=" Quit         ":gosub ta
1310 gosub ze
1320 if x>106 and x<217 and y>-15 and y<0 then gosub UNTERPRG 1
1330 if x>106 and x<217 and y>4 and y<19 then gosub UNTERPRG 2
1340 if x>106 and x<217 and y>23 and y<38 then gosub UNTERPRG 3
1350 if x>106 and x<217 and y>42 and y<57 then gosub UNTERPRG 4
1360 if x>106 and x<217 and y>61 and y<76 then gosub UNTERPRG 5
1370 if x>106 and x<217 and y>80 and y<95 then gosub back
1380 goto 960
1390 # Vierte freibelegbare und erweiterbare Hauptunterroutine! #
1400 show:
1410 gosub hi:xp=166
1420 yp=-2:t$=" UNTERPRG 006":gosub ta
1430 yp=17:t$=" UNTERPRG 007":gosub ta
1440 yp=36:t$=" UNTERPRG 008":gosub ta
1450 yp=55:t$=" UNTERPRG 009":gosub ta
1460 gosub ze
1470 if x>166 and x<277 and y>-15 and y<0 then gosub UNTERPRG 6
1480 if x>166 and x<277 and y>4 and y<19 then gosub UNTERPRG 7
1490 if x>166 and x<277 and y>23 and y<38 then gosub UNTERPRG 8
1500 if x>166 and x<277 and y>42 and y<57 then gosub UNTERPRG 9
1510 goto 960
1520 # Fuenfte freibelegbare und erweiterbare Hauptunterroutine! #
1530 options:
1540 gosub hi:xp=227
1550 yp=-2:t$=" OPTIONS 001 ":gosub ta
1560 yp=17:t$=" OPTIONS 002 ":gosub ta
1570 yp=36:t$=" OPTIONS 003 ":gosub ta
1580 gosub ze
1590 if x>227 and x<338 and y>-15 and y<0 then gosub OPTION 1
1600 if x>227 and x<338 and y>4 and y<19 then gosub OPTION 2
1610 if x>227 and x<338 and y>23 and y<38 then gosub OPTION 3
1620 goto 960
1630 # Zurueck ins Basicsystem! #
1640 back:
1650 gosub hi:gosub re1:gosub ze
1660 if x<358 or x>418 or y<165 or y>182 then return
1670 gosub hi
1680 xp1=-1:yp1=-40:xp2=638:yp2=-20:gosub re:xp=23:yp=-23:t$="Desk":gosub ta
1690 xp=70:t$="File":gosub ta:xp=120:t$="Run":gosub ta:xp=160:t$="Edit"
1700 gosub ta:xp=210:t$="Help":gosub ta:xp=500:t$="INTERSOFT"+chr$(190)
1710 gosub ta:poke systab+24,0:gosub mein:end
```

»GEM-Menü« macht die Programmierung von Pull-down-Menüs im Profilook kinderleicht

Schätze im Verborgenen

Der Basic-Interpreter Ihres QL hat mehr Befehle in petto, als Sie denken. Wir haben für Sie noch einige aufgespürt.

Im Betriebssystem des QL schlummern Befehle, die jeden SuperBasic-Programmierer interessieren. Nur, das Handbuch erwähnt sie mit keiner Silbe. Wurden diese Befehle schlicht vergessen, oder enthalten sie noch Fehler, so daß Sinclair absichtlich nicht darauf hinweist? Vielleicht handelt es sich auch schlichtweg um Gedankenlosigkeit, denn beim Experimentieren damit traten jedenfalls keine Fehler auf.

Die Befehle WHEN und END WHEN dienen zum Überwachen von Variablen. Die anderen neuen Befehle unterstützen die Fehlerbehandlung. Die Syntax der Befehle lautet:

```
WHEN <Bedingung>
<Statement>
END WHEN
```

<Statement> steht dabei für beliebig viele SuperBasic-Anweisungen. <Bedingung> muß eine bestimmte Form erfüllen: Auf der linken Seite, also direkt hinter dem WHEN, darf nur eine Variable stehen. Ein Programm darf unbegrenzt viele solche WHEN-Konstruktionen enthalten. Im Gegensatz zu einer Prozedur oder Funktion, die sofort nach der Eingabe zur Verfügung steht, muß die Zeile mit dem WHEN-Befehl erst ausgeführt werden.

Mit der WHEN-Prozedur kann man Variablen überwachen. Das gilt immer für die Variable, die links von der Bedingung steht. Sobald die angegebene Bedingung erfüllt ist, wird die entsprechende WHEN-Prozedur ausgeführt. Es dürfen für jede Variable mehrere WHEN-Prozeduren aktiv sein.

Alle Variablen, die überwacht werden, trägt das System in eine Tabelle ein. Sobald eine dieser Variablen einen neuen Wert zugewiesen bekommt, testet das System die WHEN-Bedingung auf wahr oder unwahr. Ein INPUT gilt dabei nicht als Zuweisung. Man kann sich aber durch einen Trick helfen:

```
INPUT a:a=a
```

Nur globale Variablen lassen sich überwachen; ein Schachteln von WHEN-Prozeduren ist erlaubt. Nach Abarbeitung der Prozedur springt das Programm an die entsprechende Stelle zurück, an der die Bedingung erfüllt wurde, und führt das folgende Statement aus. Listing 1 zeigt einige einfache Beispiele für die Verwendung von WHEN.

Kompaß für verirrte Variablen

Das WHEN ERRor ähnelt sehr dem WHEN. Nur gilt hier die Bedingung als wahr, also ausführbar, wenn ein Fehler aufgetreten ist. Was alles als Fehler zählt, ersehen Sie aus der Tabelle. CTRL-Leertaste (Fehlermeldung: Abgebrochen) faßt das Programm nicht als Fehler auf. Tritt nun einer der 21 Fehler auf, springt das Programm in die dafür vorgesehene Prozedur. Zur Fehlererkennung stehen die Funktionen ERNUM und ERLIN zur Verfügung. ERNUM liefert die Fehlernummer (Tabelle) und ERLIN die Zeilennummer der Zeile, in der der Fehler auftrat.

Der Befehl REPORT bewirkt die Ausgabe einer Fehlermeldung. Vorgesehen ist dazu der Kanal 0, aber es kann jeder Kanal (»REPORT #Kanalnummer) diese Funktion überneh-

```
100 REMark ################################
110 REMark  Demo für WHEN
120 REMark  R.W. Gerling
130 REMark  Juni 1986
140 REMark ################################
150 WHEN neu=rechts
160 neu=links+1
170 bereich=bereich+1
180 mark bereich
190 END WHEN
200 REMark ################################
210 WHEN neu=links
220 neu=rechts-1
230 bereich=bereich-1
240 mark bereich
250 END WHEN
260 REMark ################################
270 DEFine PROCedure mark(bereich)
280 AT 11,links:PRINT bereich,
290 AT 11,rechts:PRINT bereich+1,
300 END DEFine
310 REMark ################################
320 links=10
330 rechts=30
340 bereich=0
350 alt=20:CLS
360 AT 10,links:PRINT "I"
370 AT 10,rechts:PRINT "I"
380 mark bereich
390 AT 10,alt:PRINT"*"
400 REPeat main
410     neu=alt+RND(-1 TO 1)
420     AT 10,alt:PRINT" "
430     AT 10,neu:PRINT"*"
440     alt=neu
450 END REPeat main
```

Listing 1. Ein Stern wandert zwischen den beiden »I«. Die beiden WHEN-Prozeduren sorgen für das korrekte Verhalten beim Erreichen des Randes.

```
100 REMark ################################
110 REMark  Demo für WHEN ERRor
120 REMark  R.W. Gerling
130 REMark  Juni 1986
140 REMark ################################
150 WHEN ERRor
160 fehler=ERNUM
170 SELect ON fehler
180 ON fehler=-18
190   IF zahl>0 THEN
200      PRINT#0;zahl;" ist leider zu groß"
210   ELSE
220      PRINT#0;zahl;" ist leider zu klein"
230   END IF
240 ON fehler=-17
250   PRINT#0;"Bitte nur gültige Zahlen eingeben"
260   PRINT
270   RETRY
280 ON fehler=REMAINDER
290   PRINT#0;"Den Fehler kenne ich nicht"
300 END SELect
310 END WHEN
320 REMark ################################
330 CLS:CLS#2:CLS#0
340 REPeat main
350   fehler=0
360   INPUT"Gib bitte eine Zahl ein: ";zahl
370   exp_zahl=EXP(zahl)
380   IF fehler THEN GO TO 400
390   PRINT"EXP(";zahl;") = ";exp_zahl
400 END REPeat main
```

Listing 2. Die Funktion WHEN ERRor erleichert das Fehlerabfangen sehr

men. Die Funktion ERR__NN ergibt keinen Sinn. Sie liefert im Normalfall den Wert Null, oder, falls der entsprechende Wert aufgetreten ist, den Wert Eins. Diese Information vermittelt auch ERNUM, und zwar einfacher.

Während der Fehlerbehandlung wird ein Flag gesetzt. Solange verursacht ein neuer Fehler die Fehlermeldung »IN BEARBEITUNG« und das Programm stoppt. Auf drei Wegen kann man nach der Bearbeitung der Fehler-Prozedur das Fehlerflag zurücksetzen, nämlich mit END WHEN, RETRY und CONTINUE. Verwendet man den Befehl RETRY, sollte man ganz sicher sein, daß die Fehlerbedingung beseitigt ist, sonst produziert man eine Endlosschleife. END WHEN am Ende der Fehlerbehandlung bewirkt das gleiche wie CONTINUE.

Man kann also nicht im Falle eines Fehlers aus der Fehler-Prozedur an eine beliebige Stelle im Programm springen. Entweder das Programm wiederholt das Statement oder es begibt sich zum folgenden Statement. Aus diesem Grund muß die Fehler-Prozedur die Fehlersituation beseitigen. Listing 2 zeigt ein Beispiel für die Verwendung von WHEN ERRor. Das kleine Programm demonstriert, wie einige typische Fehler abzufangen sind. Durch diese kleinen Utilities schreiben Sie Ihre Programme schneller und mit weniger Zeitaufwand.

Viel Spaß! (Rainer Gerling/hb)

Nummer	Kennung	Bedeutung	
-1	NC	Abgebrochen	Not complete
-2	NJ	Fehlerhafter Job	Invalid Job
-3	OM	Speicherüberlauf	Out of Memory
-4	OR	Bereichsüberlauf	Out of Range
-5	BO	Puffer voll	Buffer full
-6	NO	Kanal nicht geöffnet	Channel not open
-7	NF	Nicht gefunden	Not found
-8	EX	Existiert bereits	Already exists
-9	IU	In Bearbeitung	In use
-10	EF	Dateiende	End of file
-11	DF	Datenträger voll	Drive full
-12	BN	Ungültige Bezeichnung	Bad name
-13	TE	Übertragungsfehler	Xmit Error
-14	FF	Formatfehler	Format failed
-15	BP	Ungültiger Parameter	Bad Parameter
-16	FE	Fehlerhafter Datenträger	Bad or changed medium
-17	XP	Fehler im Ausdruck	Error in expression
-18	OV	Überlauf	Overflow
-19	NI	Nicht implementiert	Not implemented
-20	RO	Nur Lesen	Read only
-21	BL	Syntaxfehler	Bad line

Die Tabelle zeigt die Fehlertexte zu den Fehlernummern und Kürzeln für die Funktion ERR__NN. Die Kürzel ergeben sich aus den englischen Fehlermeldungen.

Gut gedruckt, Amiga

Möchten Sie einen Drucker mit Centronics-Schnittstelle an den Amiga anschließen? Vorsicht, wenn Sie kein spezielles Kabel verwenden, spielen Computer und Drucker verrückt!

Beim Amiga hielten sich die Konstrukteure bei der Belegung des Parallelports leider nicht an die Centronics-Norm.. Am Pin 23 liegt nicht Masse, sondern eine Spannung von 5 Volt an. Wenn man nun einen Drucker durch ein handelsübliches Kabel anschließt, ist ruckzuck ein Kurzschluß da. Sowohl der Amiga als auch der Drucker können dabei einen Schaden davontragen. Daher ist ein besonderes Verbindungskabel vonnöten, das sich ein in Maßen geschickter Bastler selbst löten kann. Alles, was man braucht, ist:
- ein Stecker im DB-25-Format
- ein Centronics-Stecker
- ein mindestens 23adriges Kabel

Diese gängigen Bauteile bekommen Sie im Fachhandel oder in Bastlerläden. Achten Sie beim Kauf eines Centronics-Steckers auf einen Stecker mit Klemm-Schneide-Prinzip. Das erspart Ihnen viel Lötarbeit. Vergessen Sie dabei nicht, daß Sie für diesen Stecker ein Flachbandkabel mit mindestens 1,27 mm Ader-Durchmesser brauchen.

Die Pins der beiden Stecker verbinden Sie anhand der Tabelle. Lassen Sie sich nicht durch die große Anzahl der Anschlüsse am Centronics-Stecker verwirren. Die Pins sind beschriftet, was Ihnen die Orientierung erleichtert. Da die Anschlüsse eng beieinander liegen, gehen Sie bitte sehr sorgfältig vor, damit keine Lötbrücken entstehen. Besonders am erwähnten Pin 23 hätte das fatale Folgen. Überprüfen Sie Ihre Arbeit auf jeden Fall mit einem Meßgerät, bevor Sie einen Test mit dem Drucker wagen!

Wenn nun das Verbindungskabel funktioniert, müssen Sie im Druckermenü des Amiga den richtigen Druckertreiber auswählen. Klicken Sie dazu im »Preference«-Programm Ihrer Workbench die Funktion »Change Printer« an. Über die Maus lassen sich alle vordefinierten Druckertreiber anzeigen und auswählen. Wenn Sie einen grafikfähigen Drucker besitzen, können Sie mit »Grafic Select« die Parameter »Shade« (Schattierung), »Aspect« (Orientierung des Ausdrucks), »Image« (Erscheinung) und »Treshold« (Schwarzweiß-Grenze) einstellen.

Beim Anschluß eines Farbdruckers genügt es nicht, nur den Druckertyp anzugeben. Der Amiga will auch »wissen«, daß Sie damit farbig drucken.

Mit »Aspect« wählen Sie eine horizontale oder vertikale Lage des Bildes. Bei Querdruck fallen die Hardcopies größer aus als bei horizontalem Druck. Möchten Sie die Grafik invertiert ausgeben, stellen Sie bei »Image« einfach »Negative« ein.

Die Skala am oberen Bildschirmrand interessiert nur bei Schwarzweiß-Druck. Je kleiner der eingestellte Wert, desto heller wird das Bild.

Um Hardcopies von Bildern zu erzeugen, benötigen Sie ein geeignetes Grafikprogramm wie Deluxe Paint oder Graphicraft. Hier als Druckertest ein kleiner, aber sehr hilfreicher Trick: Um das Inhaltsverzeichnis einer Diskette auszugeben, verwenden Sie folgenden Befehl im CLI-Modus:
»DIR >PRT:«

Sie können jede Bildschirmausgabe auf den Drucker umleiten, sobald Sie an den Befehl »>PRT:« anhängen. Jede Textdatei wird durch: »TYPE filename TO PRT:« ausgedruckt.

(Ottmar Röhrig/gn)

Parallel-Port Amiga	Centronics-Stecker
1-13	1-13
14	19
15-22	20-27
25	31

Ein Programm, das wirklich löscht

Datenschutz heißt die Devise für dieses Programm. Wenn Sie eine Datei mit »Filekill« löschen, ist sie im elektromagnetischen Jenseits, im Gegensatz zum Standard-Löschen mit »Maus« und »Papierkorb«.

Beim Löschen einer Datei ist jeder Computer wesentlich schneller als beim Schreiben. Das hat folgenden Grund: Der Computer löscht die Datei nicht wirklich, also physikalisch, sondern er kennzeichnet den Eintrag im Inhaltsverzeichnis und die Sektoren nur als leer. Mit einem Diskettenmonitor und Kenntnissen über die Verwaltung einer Diskette ist es einfach, die Daten wieder zu reaktivieren. Man setzt die Gelöscht-Kennung zurück, und alle Daten sind wieder vom Betriebssystem lesbar.

Das ist hilfreich beim versehentlichen Löschen von Daten. Manchmal möchte man aber, daß die Daten absichtlich nicht mehr reaktivierbar sind. Genau dafür ist das Programm da.

Nach dem Durchlauf dieses Programms finden Sie garantiert an keiner Stelle mehr etwas von den alten Daten, sogar der Dateiname ist auf Nimmerwiedersehen von der Diskette verschwunden. Es handelt sich um ein TOS-Programm, an das der Name der zu löschenden Datei übergeben werden muß. Nach dem Eintippen und Compilierungslauf müssen Sie die Endung auf ».TTP« ändern. Es spielt keine Rolle, ob sich die Datei in einem Ordner befindet oder nicht.

Beispiele für Eingaben sind:

adressen.dat

Löscht die Datei, wenn sie sich auf demselben Laufwerk und in demselben oder in keinem Ordner befindet.

a:\adressen.dat

Löscht die Datei, wenn sie sich auf Laufwerk A und in keinem Ordner befindet.

a:\geheim\adressen.dat

Löscht die Datei, wenn sie sich auf Laufwerk A und in dem Ordner »GEHEIM« befindet. (Michael Schutte/hb)

```
#include <osbind.h>

int buf[512];                                    /* 1k Puffer */

char diskbuf[44];

long ld;                                         /* Länge der Datei */

main(argc,argv)                                  /* .TTP Programm, Dateiname wird an */
                                                 /* das Programm übergeben */
int argc;
char *argv[];
{
 register int i;                                 /* Laufvariable */
 int fh,err;                                     /* Filehandle, Fehler */
 char neu[40];                                   /* Neuer Dateiname */
 long bytes=0,sc=0;                              /* zum Lesen und Schreiben */
 for(i=0;i<=512;buf[i++]=0);                     /* Puffer löschen */
 argc--;
 if (argc==0)
  { printf("Bitte einen Dateinamen angeben !\n");
    goto schluss;
  }
 Fsetdta(diskbuf);                               /* Disk-Übertragungsadresse setzen */
 printf("Lösche %s\n",argv[1]);
 fh=Fopen(argv[1],2);                            /* Öffnen */
 if (fh<0)                                       /* Vorhanden ? */
  { printf("Kann ich nicht finden !\n");
    goto schluss;
  }
 ld=Fseek(0L,fh,2);                              /* Dateilänge ermitteln */
 Fseek(0L,fh,0);                                 /* Zum Anfang... */
 do
  { sc=ld-bytes;
    if (sc>1024) sc=1024;
     err=Fwrite(fh,sc,buf);                      /* Leeren Puffer schreiben */
```

»Filekill« löscht Daten wirklich auf Nimmerwiedersehen

```
    if (err<0)                          /* mit Erfolg ? */
     { printf("Schreibfehler !\n");
       Fclose(fh);
       goto schluss;
     }
    bytes+=err;
  }
  while(bytes<ld);
  printf("\n");
  err=Fclose(fh);                      /* Schließen */
  if (err<0)                            /* Geglückt ? */
   { printf("Kann nicht schließen !\n");
     goto schluss;
   }
  strcpy(neu,argv[1]);                 /* Dateinamen im Directory */
                                       /* unkenntlich machen */
  i=strlen(neu);
  while(neu[i]!='\\' && neu[i]!=':' && i>=0) i--;
  if (i==0) i--;
  neu[++i]=0;
  strcat(neu,"XXXXXXXX.XXX");
  Frename(0,argv[1],neu);
  err=Fdelete(neu);                    /* ...und aus dem Directory löschen */
  if (err<0) printf("Nicht korrekt gelöscht !\n");
  schluss:
  printf("\nOk. (Taste)\n");
  gemdos(8);                           /* Auf Taste warten */
}
```

»Filekill« (Schluß)

Effekthascherei

Diese Sammlung von Titel-Effekten für den Atari ST mit Farbmonitor würdigt jeder Grafik-Freak, der neben der Zeichenkunst auch das Programmierhandwerk einigermaßen beherrscht, als ein nützliches Hilfsmittel.

Selbstgemalte oder fertige Grafiken in seine Programme einzubinden, das wünscht sich so mancher Programmierer. Egal, ob es sich bei seinem Favoriten um Basic, C, Forth oder Assembler handelt – nachfolgend abgedruckte Programm-Module versetzen jedermann, der einen Assembler sein eigen nennt, in die glückliche Lage, Grafiken in seine Programme einzubauen. Einige nette Effekte sorgen zudem für ein wenig Abwechslung.

Bei den bereits erwähnten Modulen handelt es sich um drei Effekt-Module und um ein universelles Grundprogramm, das hinter jedes Effekt-Modul angehängt wird. Das Grundprogramm enthält Unterroutinen, auf die jedes Effekt-Modul zugreift, wie zum Beispiel Löschen des Bildschirms oder Laden der Grafik von Diskette. Bei den Modulen handelt es sich im einzelnen um folgende Effekte:

1. Herabrollen der Grafik, wobei ein gewisses »Rollo-Feeling« aufkommt.
2. Aufbauen der Grafik durch viele kleine, quadratische Ausschnitte.
3. Weiches Einblenden der Grafik.

Nun noch ein paar Hinweise zur Programmeingabe. Es empfiehlt sich, jedes Modul separat einzugeben und zu speichern. Später wird der »Rumpf«, das Grundprogramm, mit je einem Grafikteil zusammengebunden. Dabei ist darauf zu achten, daß das Effekt-Modul an erster Stelle steht, unmittelbar gefolgt vom Grundprogramm.

Das Grundprogramm liest in der abgedruckten Version Grafiken des Malprogramms »Neochrome« ein. Ein paar Änderungen passen das Programm aber auch an »Degas« an. Diese lauten wie folgt:

Zeile ändern:

```
MOVE     #4,D0  ändern in  MOVE    #2,D0
(5. Zeile ab INIT)
```

Zeilen streichen:

```
MOVE.L   #FILLER,A0
MOVE     #92,D0
BSR      READ
(Zeilen 10 bis 12 ab INIT).
```

Sämtliche Module wurden mit dem Seka-Assembler geschrieben. Bis auf wenige Änderungen läuft der Code aber auch auf anderen Assemblern. Ein Hinweis noch für die Seka-Anwender: Vor dem Starten des Programms vom internen Debugger aus muß der Userstackpointer (USP) einmal um rund \$100(h) Byte herabgesetzt werden, da es sonst zu einem Bus-Error kommt; Objekt-Files, die man vom Desktop lädt, laufen jedoch einwandfrei.

Anwendungen für diese Effekte finden sich sicherlich genug. Das Grundprogramm und die drei Effekt-Module finden Sie auf der Leser-Service-Diskette.

(Carsten Reinhardt/ts)



Inserentenverzeichnis

Das nächste



Sonderheft

erscheint am 31. 10. 1986 zum Thema Spiele

Anzeigenschluß ist der 30. 09. 1986

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger (sc)
Stellv. Chefredakteur: Michael Lang (lg)
Redakteure: Gregor Neumann (gn), Horst Brandl (hb), Heinrich Lenhardt (hl), Johannes Leckebusch (le), Toni Schwaiger (ts)
Chef v. Dienst: Petra Wängler
Schlußredaktion: Eva Hierlmeier
Redaktionsassistenz: Rita Gietl (289)

Fotografie: Jens Jancke

Titelgestaltung: Heinz Rauner Grafik-Design

Layout: Leo Eder (Ltg.), Sigrid Kowalewski (Cheflayouterin) Rolf Raß, Katja Milles

Produktionsleiter: Klaus Buck (180)

Anzeigenverkaufsleitung: Ralph-Peter Rauchfuss

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. (042) 41 56 56, Telex: 862329 mut ch
USA: M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; Tel. 415-366-3600, Telex 752-351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Anzeigenverkauf: Britta Fiebig (211), Helmut Distl (398)

Anzeigenverwaltung und Disposition: Patricia Schiede (172)

Marketingleiter: Hans Hörl (114)

Vertriebsleiter: Helmut Grünfeldt (189)

Verlagsleiter M&T Buchverlag: Günther Frank (212)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Str. 96, 7000 Stuttgart 1, Tel. (07 11) 6483-0

Bezugsmöglichkeiten: Leser-Service: Telefon (089) 4613-249. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Bezugspreis: Das Einzelheft kostet DM 14,–

Druck: SOV St. Otto-Verlag GmbH, Laubanger 23, 8600 Bamberg

Verantwortlich: Für redaktionellen Teil: Michael Scharfenberger
Für Anzeigen: Britta Fiebig.

Redaktionsdirektor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0, Telex 5-22052

Telefon-Durchwahl im Verlag:
Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen (089) 4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.





MC68020

Der 32-Bit-Prozessor!

Literatur zur M68000-Familie:

Teil 1, Grundlagen und Architektur sowie Beschreibung der Adressierungsarten und des Befehlssatzes des M68000, 550 Seiten, DM 79,–.
Teil 2, Programmierbeispiele sowie Beschreibung der M68000-Familie bis hin zum 68020, inkl. Single-Board-System auf 68000-Basis, 350 Seiten, DM 69,–.
Beide Bände sind im Fach-Buchhandel erhältlich.



DIE LEGENDÄREN FÜR ATARI 520 ST & AMIGA



BORROWED TIME

„Sam, you're a dead man." Das Telefon klingelt. Eine ersterbende Stimme warnt Dich. Du denkst nach. Da sind rund 20 undurchsichtige Typen, von dene jeder genug Gründe hat, Dich umzubringen. Doch wer ist der Mörder? Du hast nur eine Chance: den Killer zu finden, bevor er Dich findet. Deine Zeit ist nur geliehen — die Uhr läuft ab.

Technische Besonderheiten

- Über 70 Bilder in hochauflösender Grafik und teilweiser Animation.
- Help Modus — gibt Hilfen, ohne die Lösung zu verraten.
- Einfachste Bedienung durch Windows und Menüs.
- Großzügige Kommunikation durch umfangreichen Wortschatz.
- Mit Demo-Programm

Ein fantastisches Text-Grafik-Adventure voller Spannung.



HACKER

Ganz zufällig geraten Sie in ein völlig fremdes Computersystem.

Was nun?

Log on
Nur dieses eine kleine Wort ist auf dem Bildschirm zu sehen. Wie geht's nun weiter? Das Passwort ist nicht bekannt. Der Firmenname auch nicht. Aber als ordentlicher Hacker werden Sie dies schon herauskriegen. Es gibt keine Anleitung. Keine Regeln. Keine Hinweise. Sie sind ganz auf sich selbst angewiesen. Den Weg in das Computersystem haben Sie zufällig gefunden. Finden Sie auch wieder hinaus?

Eine total neue Spielidee!



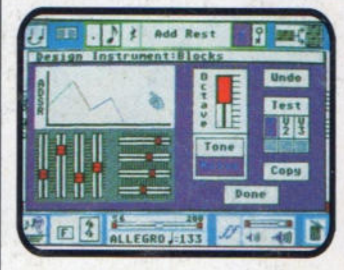
MINDSHADOW (AMNESIE)

Wer bist du? Wo bist du? Was wirst du tun?

Ein fantastisches Text-Grafik-Adventure voller Intrigen.

Du kannst Dich an nichts mehr erinnern.
Du befindest Dich an einem wüstenähnlichen Strand. Aber an welchem?
Du willst Deine Identität um jeden Preis herausfinden.
Eine Entdeckungsreise führt Dich rund um die Welt und bringt Dich der Wahrheit immer näher. Einer Wahrheit voller Intrigen und Gefahren. Mit Hilfe des mysteriösen Condors kommst Du dem Verräter immer näher . . .

Fantastische hochauflösende Grafik mit über 80 Bildern.



MUSIK STUDIO Designed by Audio Light

Mit dem Music Studio verfügen Sie über ein komplettes Orchester und haben die Instrumente quasi in Ihren Fingerspitzen. Auch ohne Programmierkenntnisse oder musikalische Erfahrungen komponieren, arrangieren, korrigieren und betexten Sie Ihre eigenen Stücke. Entdecken Sie die vielfältigen Möglichkeiten eines richtigen Synthesizers — Sie werden überrascht sein.

- Maus gesteuert
- 3 Oktaven / 3-stimmig
- Incl. Demo-Programm für alle Musik-Neulinge
- Abspeichern, Laden und Ausdrucken jederzeit möglich
- Incl. diverser vorproduzierter Melodien
- Bis zu 15 Instrumente pro Titel
- Midi-kompatibel



NEU

ACTIVISION
HOME COMPUTER SOFTWARE

ACTIVISION Deutschland GmbH. Postfach 76 06 80, 2000 Hamburg 76.
VERTRIEB DEUTSCHLAND: Ariolasoft (Exclusiv-Distributor), Rushware (Autorisierter Mitvertrieb) VERTRIEB ÖSTEREICH: Karasoft (Exclusiv-Distributor) VERTRIEB SCHWEIZ: HILCU (Exclusiv-Distributor)